SentraWare

Administrator's Guide

SentraWare

Administrator's Guide

Version 1 Release 1 Modification 0

Table of Contents

Chapter 4 - Managing the System

Chapter 5 - Working with Actions and Scheduler

List of Figures

List of Tables

Syntax and Conventions

This document uses syntax and conventions designed to help you understand the material provided. Special syntax and conventions are described below.

A grouping of text in this document that appears on a display monitor (from a single command line to an entire screen image) is shown in a distinct type style within a drop-shadowed box; for example:

```
//TSPSTEP1 JOB (account),'RESTORE GENLIB'
//STEP1    EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=*
//TAPEIN   DD  DSN=OMC.V1R0M1.GENLIB,
//             DISP=OLD,
//             UNIT=TAPE,
//             VOL=SER=OMC101,
//             LABEL=(1,SL,EXPDT=98000)
//DISKOUT  DD  DSN=prefix.GENLIB,
//             DISP=(NEW,CATLG,DELETE),
//             UNIT=unit,
//             VOL=SER=volser,
//             SPACE=(6160,(105,50,8))
//SYSIN    DD  *
```

Small amounts of text (a few lines) you type at your workstation are shown in an box; for example:

```
TSP      IATYDSD  PRTY=5,REENT=NO,XABLE=YES,
                  NOREQ=0,DRVR=IATTSP,PABLE=NO,
                  MAXCT=1,MUCC=NO
```

A text line that exceeds the right boundary of the box is split and is continued on one or more indented lines. Commands that appear in this document as multiple lines are entered at the keyboard as a single line.

Capital letters indicate literal text. Lower-case letters indicate a variable field for which a value must be supplied.

Two or more possible values for a field are enclosed within a pair of braces, { }, and separated by a vertical bar, |. Only one of the listed values may be specified. If there is a default value, it will be underlined.

Optional text is shown enclosed within a pair of brackets, [ ].

Text within a paragraph that requires special attention is printed in bold and/or italic:

This is an example of ***important text*** within a paragraph.

Special information that is vital to the proper operation of this product will be set off as a note:

NOTE: ***This is a note; it is used to draw your attention to a particularly important aspect of the subject matter.***

You may encounter other text not directly related to the technical information in this document-but which has been included for your benefit. This text is formatted like the following sample:

Occasionally, you will see text formatted this way. This text provides general (nontechnical) information you may find useful.

Trademarks

The following are trademarks of International Business Machines Corporation:

- IBM
- TSO
- ISPF/CICS
- MVS, MVS/ESA, MVX/XA/RACF
- VTAM/JES/328X

The following are trademarks of Tone Software Corporation:

- CompuLert
- SentraWare

Other organization brand and product names mentioned are registered, trademarked, or service marked by their respective companies or holders.

Chapter 1 - Introduction

This document describes the specialized procedures and activities that the *SentraWare* (SWA) System Administrator must perform. The operational procedures and activities the *SentraWare* system commands described in this document are defined in the *SentraWare Reference Manual*.

The System Administrator is the *SentraWare* user who is responsible for the overall operation of the *SentraWare* system. Users with **ADMIN** permission are allowed to modify static *SentraWare* configurations using a series of **rc** commands (**rc** stands for reference change). System Administrators are the only users with **ADMIN** permissions.

Administrative Functions

The System Administrator may execute all *SentraWare* system commands. The following *SentraWare* functions are restricted to the System Administrator, unless otherwise noted.

- Creating *SentraWare* centers.
- Defining and assigning *SentraWare* channels.
- Assigning logins (user names and passwords) for other *SentraWare* users.
- Assigning permissions for individual *SentraWare* users.
- Controlling user programming activity.
- Defining the automation, managing actions.
- Defining and controlling the scheduler execution.
- Tracking system resources and performing resource measurements.
- Defining system-wide parameters.
- Maintaining dial-in/dial-out modems.
- Scheduling and interpreting administrative reports.
- Evaluating data base integrity and scheduling backups.
- Monitoring resource usage.

The following is a list of commands are restricted to the System Administrator:

- all **rc-** commands
- **allow-system**
- **inhibit-system**
- **archive**

Occasionally, the System Administrator shares permission to use a command with other users, such as a Center Administrator. In this case, the System Administrator elects to perform the associated functions, or delegate the responsibility to other users sharing the command permission. The permission may only be valid for a specified entity or center.

Using this Manual

This manual contains the following chapters:

- **Chapter 1** (this chapter) introduces the *SentraWare* system, describes the components of *SentraWare* and the job each of these components performs.
- **Chapter 2** describes the procedures to install *SentraWare*; the planning and installation processes.
- **Chapter 3** describes the administration of *SentraWare* users - how to assign login, password and permissions.
- **Chapter 4** describes the administrative tasks such as defining a channel, adding centers and entities.
- **Chapter 5** describes the procedures to build automatic response actions and scheduler.

Introduction to the *SentraWare* System

SentraWare centralizes and automates the operations, maintenance, and support personnel functions associated with telecommunications operations centers. The system provides features to make these functions more efficient and effective.

The following terms have specific meanings within the context of *SentraWare*:

Term	Application
center	A logical grouping of entities as defined by the System Administrator. Systems included in a center need not occupy the same physical location.
entity	Designates a device capable of providing a translatable byte stream to *SentraWare* through an RS-232, datakit, or network connectivity. Normally, an entity is a telecommunications system whose that provides ASCII output. *SentraWare* typically uses splitters to capture console output over an RS-232 connection.
channel	Bi-directional channel of data between *SentraWare* and an entity that is controlled by a **DAP** (Data Acquisition Process).
session	The process in which an operator interacts with an entity to accomplish a specific task.
users	Indicates personnel who interface with *SentraWare*.
action	A pre-recorded response to a SentraWare detected or timed event.
Log	An area in the database containing messages and alarms for an entity.
pattern	The criteria for selecting messages and alarms.

Table 1.1 - SentraWare Specific Terms

SentraWare System Features and Functionality

The "tools" and system features *SentraWare* provides include:

Feature	Description
Logging	Records data output from an entity. All messages logged are time-stamped and placed into a disk logging area. Messages may also be archived and viewed with one of *SentraWare's* browsing programs from a workstation. With the **playback** feature, messages may be viewed the same as if displayed in the console terminal.
Console Access	Allows the user to access a channel through the entity connected to *SentraWare*. For example, if the entity is connected to *SentraWare* using an RS-232 line connected to the entity's console, accessing the entity provides access to the system console, even in single-user mode.
Automatic Response (actions)	An optional set of commands executed or called by the Scheduler. Each action is identified by an action ID and is placed into an action database.
Scheduler	Allows the user to create command lists called Scheduler jobs.

Table 1.2 - SentraWare System Tools and Features

SentraWare software modules are described in the following table.

Software Module	Description
Executor	**Executor** is the first *SentraWare* module started by UNIX. **Executor** checks the operability of the operating system and the database before launching the rest of *SentraWare'*s modules. **Executor** monitors all other *SentraWare* modules and requires every *SentraWare* module to send a heartbeat. If **Executor** does not receive a heartbeat in a specified time from a *SentraWare* module, and its exit code is non-fatal, it automatically tries to reactivate the troubled module. Using **Executor**, a System Administrator can shutdown or restart whole *SentraWare* modules as well as control the state of individual *SentraWare* modules. The commands associated with **Executor** are described in Chapter 4, "Managing the System".
Event Dispatcher	All the *SentraWare* modules, which update the database, notify the **Event Dispatcher** upon successful update of the database. **Event Dispatcher** consolidates all Configuration Database change requests. It has network capability and provides the ability for real-time *SentraWare* mirroring.
Channel Manager	Provides connectivity with the **DAP Manager**. It consolidates all input to an entity, requests the **DAP Manager** to open/close channels (restore/remove log).
Data Acquisition Process (DAP)	Buffers streams of data coming from monitored entities into lines and messages. May reside on the central computer subsystem or on supplemental remote processors. Executes and controls threads for every channel. Each thread controls a serial port or 'telnet'. **DAP** establishes the TCP/IP connection to the **Data Dispatcher** directly upon request from the **DAP Manager**. It consolidates all input/output for its threads. The **DAP Manager** (**dapmgr**) provides front-end access to the **DAP** subsystem for control and configuration. The **DAP Manager** starts **DAP**s and controls them. The **DAP Manager** stores all subsystem configurations locally. **DAP**s are described further in Chapter 4, *"*Managing the System*"*.
Data Dispatcher	Distributes incoming data to **Comparator**, **VT Logger**, or **Message Analyzer** based on their interests. **Data Dispatcher** provides the ability to add new processes in parallel without modification of all the others.
VT Logger	Stores the entity logging data into a database as Entity Vterm Log. It converts the incoming entity data, which contains terminal dependent escape sequences, to *SentraWare* Vterm data.
Message Analyzer	Stores the entity logging data into a database as Entity Message Log. **Message Analyzer** applies the logging pattern, formatting pattern (if any), and automatically converts any terminal dependent escape sequences in the entity data to *SentraWare* Vterm data.

Table 1.3 - SentraWare System Software Modules (Part 1 of 2)

Auto Response	Executes automation actions specified in the MA Table. **Auto Response** also executes actions specified by Scheduler. It generates an Action Log.
Scheduler	Notifies **Auto Response** about any Actions to be executed. The **Scheduler** assigns job priorities, which **Auto Response** executes, with the highest priority first. This module can: • Place scheduled jobs on a waiting list • Take jobs off the waiting list after the user-specified time limit has been exceeded • Issue a minor alarm when the user-specified time limit has been exceeded This module schedules most *SentraWare* system commands except "**rc**" commands.

Table 1.3 - SentraWare System Software Modules. (Part 2 of 2)

SentraWare Directory Structure

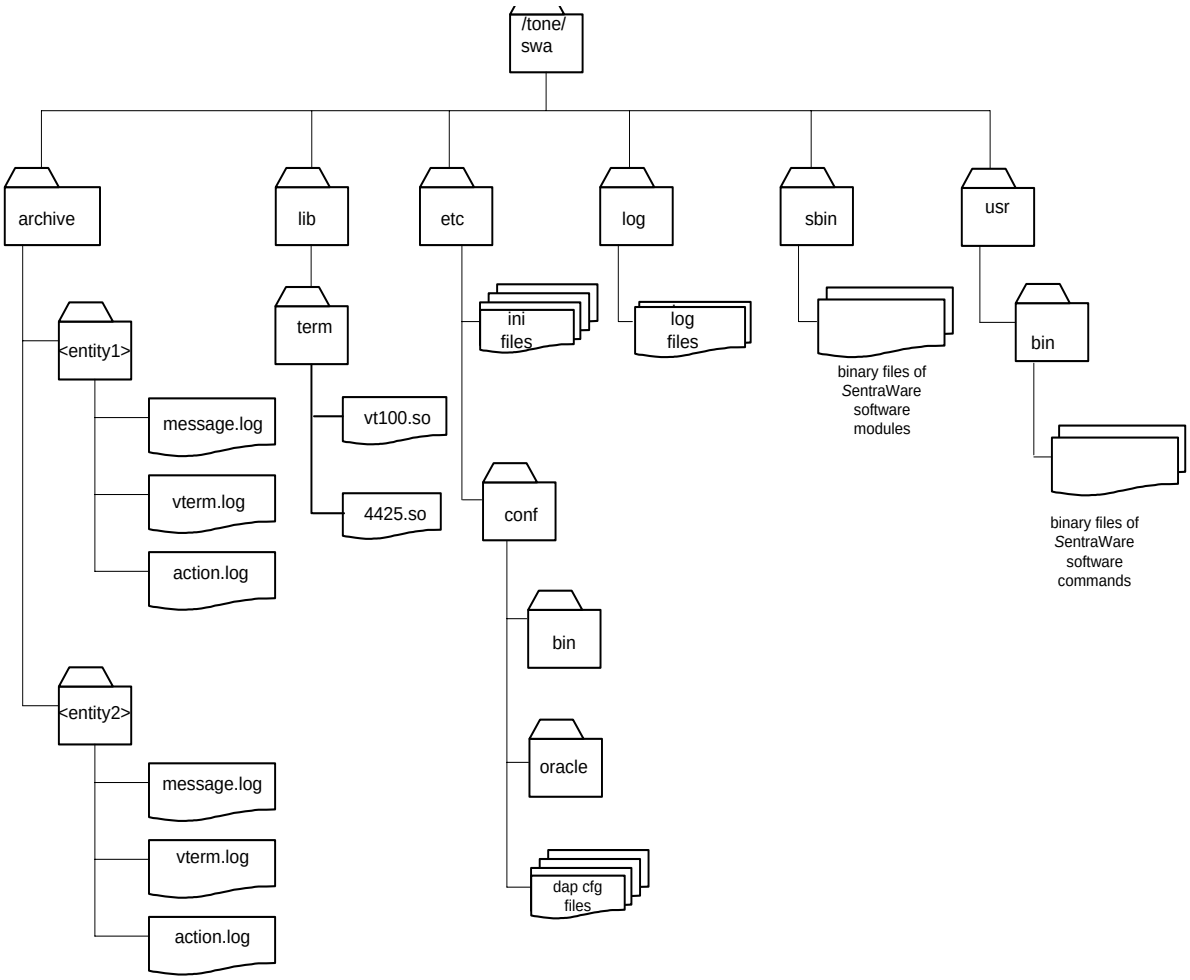


Figure 1.1 - SentraWare Directory Structure

The default *SentraWare* root directory is **/tone/swa**. This default is defined in the **/etc/SentraWare** file under the keyword **SWATOP**. The binary files of *SentraWare* software modules are stored in the **SWATOP/sbin/** directory. The binary files of *SentraWare* commands are stored in the **SWATOP/usr/sbin/** directory.

In the current release of *SentraWare*, all entity logging files, user preferences, and system definitions are stored in the database server and are not included in the *SentraWare* tree.

Working with the UNIX Shell

Before executing the *SentraWare* commands from the UNIX shell, one of two conditions must be met:

1. The user must be listed as an equivalent user (using the **rc-usr** command). *SentraWare* commands can be executed directly once the user has logged into the UNIX machine, with *SentraWare* running. Execute "**swash**" to enter the *SentraWare* command language environment. (The user is not prompted for login and password.)

2. Environment variables SWAUSER and SWAPASSWD must contain the *SentraWare* user name <username> and password, respectively.

The following is an example of setting up environment variables SWAUSER and SWAPASSWD for a user with username "**jdoe**" using the password "**ade57**".

```
$ SWAUSER=jdoe; export SWAUSER
$ SWAPASSWD=ade57; export SWAPASSWD
```

✎ ***NOTE:*** ***Set the Path to include the sub-directory where all of SentraWare's command binaries reside. The default value is in the sub-directory $SWATOP/usr/bin.***

Once SWAUSER and SWAPASSWD have been set up, the *SentraWare* commands can be executed from the UNIX shells. The user can execute **swash** to enter to the *SentraWare* command language environment and is not prompted for login and password.

Working with the Command Line (swash) Interface

In addition to all common UNIX shell languages, the *SentraWare* Command Language is also available. Enter "**swash" RETURN** at the UNIX prompt to enter the *SentraWare* Command Language. If the UNIX login name of a user <user> was specified as a response in the **equivalent** prompt in the **rc-usr**, **swash** automatically logs into the *SentraWare* database as the corresponding *SentraWare* user name <username> and presents a **swash** prompt. Otherwise **swash** prompts for login name and password.

To distinguish which language is in effect, examine the language prompt on the screen. If the **@** prompt is displayed, the *SentraWare* Command Language syntax is in effect. If the **$** (dollar sign) prompt is displayed, the language is the UNIX system shell. If the prompt is a **#** (pound sign), superuser or logged in as root is in effect.

swash allows users to create centers, entities and other components that make up *SentraWare*. Entering a question mark (**?**) at the **swash** prompt, displays a list of commands that are valid in the **swash** environment. Entering a question mark (**?**) after a command or command prompt activates help text that summarizes the information needed to continue. Entering a slash (/) after each data item provided to an interactive command prompt invokes the next prompt, if the command requires multiple pieces of information.

```
@  ?

Valid commands are:
access        inhibit-system  rc-ma          verify-catalog
acknowledge   output-access   rc-pat         verify-cent
allow-log     output-alarms   rc-sched       verify-ent
allow-system  output-stat-ent rc-usr         verify-ent-ma
archive       output-system   remove-alarm   verify-ma
bin           playback        remove-log     verify-pat
browse        rc-act          report-alarm   verify-perm
swash         rc-audname      restore-log    verify-sched
filter        rc-cent         verify-act     verify-usr
inhibit-log   rc-ent          verify-audname

@
```

*Figure 1.2 - **swash** Screen Example.*

Full descriptions of the commands listed in Figure 1.2 can be found in the *SentraWare Reference Manual*.

Command Terminators

If a *SentraWare* command with a command terminator is entered, the system searches the system files for that command. When the command is found, the system executes the requested program. A command terminator (shown by a < **t** > in the system documentation) signals the system to start the required programs. When execution is complete, the system returns the @ (at sign) prompt. The characters used to terminate are **?**, /, or the **ENTER** key. Two command terminators provide information:

Terminator	Application
/	Requests the system to prompt for missing information. If only entering half of a command line and ending it with a slash (/), the system prompts for the next option of the command line. In the following example, the command **output-system** is entered with a slash (***/***) as a command terminator. The op-system command can request more than one option. **op-system**+**ENTER** asks to display the status of all software modules in *SentraWare*. **op-system Eve**+**ENTER** asks the system to display the status of the Event Dispatcher module.

```
@ op-system/
module = Eve/
 PF

                        System Status Report
                        Thu Jan 21 16:43:45 1999

      NAME        PID  STATUS MODE STARTS   LAST START    LAST TERM.    REASON
--------------- ----- ------ ---- ------ -------------- -------------- -----
EventDispatcher 15261 ALIVE  RUN       1 01-21,11:19:53 N/A              N/A
  OK
```

ENTER	Allows the user to default all remaining keyword values on commands that have multi-level prompting. If used as a response to the first prompt at a level, returns to next higher level.
?	Requests help in the form of more detailed system responses (instead of a command prompt). Therefore, when entering the same command (**op-system**) with a question mark (**?**) as the terminator, the system responds by displaying the correct command format. For example:

Table 1.4 - Command Terminators (Part 1 of 2)

```
@op-system?
 The valid format is:

   output-system module=<modname>,...,<modname>] <t>
```

? (continued)	The following example uses the slash (**/**) to request a prompt (in this case, "module ="), then enters a question mark (**?**) for further prompting.
!	The ! (bang) symbol cancels the command in process.
^C or ^D	Cancels the command in process.

```
siska @ output-system/
module = ?
List of SentraWare modules:

        EventDispatcher
        DataDispatcher
        DapManager
        ChannelManager
        MessageAnalyzer
        VtLogger
        Comparator
        Poller
        AlarmProcessor
        AutoResponse
        Scheduler
        PatternServer

You need to type at least 3 first characters (case sen-
sitive).
Default: all modules.

module = Eve/  PF
```

Table 1.4 - Command Terminators (Part 2 of 2)

SentraWare System Responses

When entering a *SentraWare* command, the system acknowledges with a response code. This code confirms that the input has been received, traces the command's progress, or shows that input errors have been detected. The system provides two types of responses:

- acknowledgment responses
- error responses

Acknowledgment Responses

Acknowledge responses inform the user that input has been accepted by the system:

Response	Application
IP	Shows the command was accepted and is **I**n **P**rogress.
OK	Indicates the command completed successfully or that further output is immediately forthcoming.
PF	Indicates that the command was accepted and a **P**rintout will **F**ollow, after a short pause.

Table 1.5 - Acknowledgment Responses

NOTE: ***IP and OK acknowledgment responses are often used together. The IP indicates the command is being processed; the OK indicates a successful completion***

Error Responses

Error responses inform the user of error conditions:

Response	Application
?E	Indicates that an **E**rror has been detected in the user's input.
NG	Indicates that the command is **N**ot **G**ood and can not be executed or completed because of false or missing file data, or because the user specified use of unavailable system facilities.
RL	Indicates that the command cannot be executed because another user is using a related command that might interfere or the system is overloaded.

Table 1.6 - Error Condition Responses

For a complete listing of all commands and their formats, refer to the *SentraWare Reference Manual*.

Chapter 2 - Installing *SentraWare*

Installation Overview

Configure *SentraWare* in one of the following ways:

- Server-Based Configuration
- Client/Server Configuration
- Client/Server/DAP Configuration

Each configuration necessitates a different system requirement; covered individually in this chapter.

Use the following diagrams to determine the type of installation to perform.

Server-Based Configuration

Figure 2.1 illustrates a server-based installation.

- All the *SentraWare* software, the Oracle server software and the database objects are installed on one server.
- Entities are connected to the server over the network or serial lines.

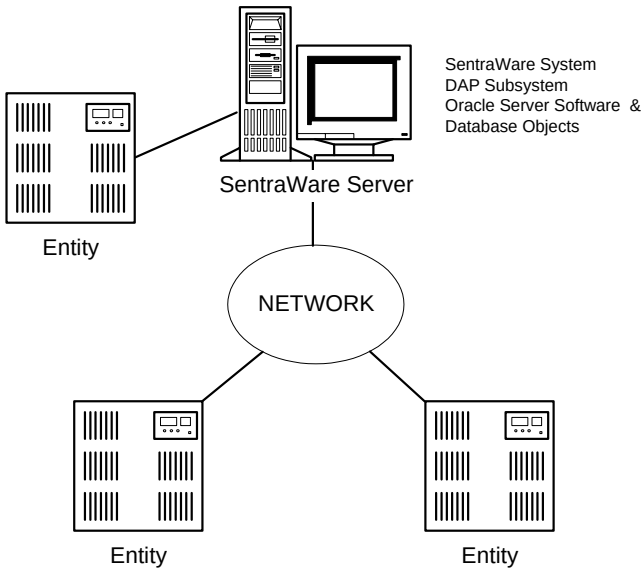


Figure 2.1 - Server-Based Configuration

Client/Server Configuration

Figure 2.2 illustrates a client/server installation.

- The Oracle server software and the database objects are installed on the database server.
- The *SentraWare* software, the DAP Subsystem, Oracle SQL*Plus and Oracle Net8 are installed on the *SentraWare* server. *SentraWare* uses Oracle Net8 as the primary interface to the Oracle database.

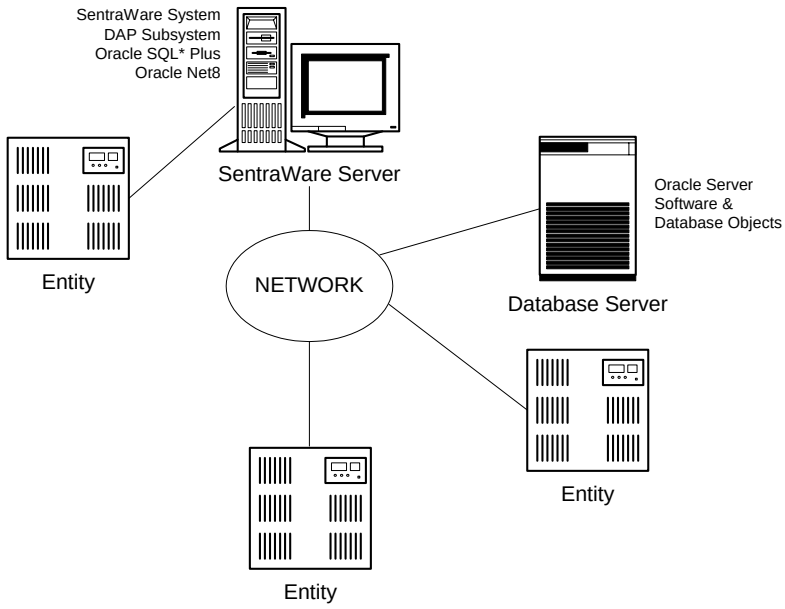


Figure 2.2 - Client/Server Configuration

Client/Server/DAP Configuration

Figure 2.3 illustrates a client/server/DAP installation.

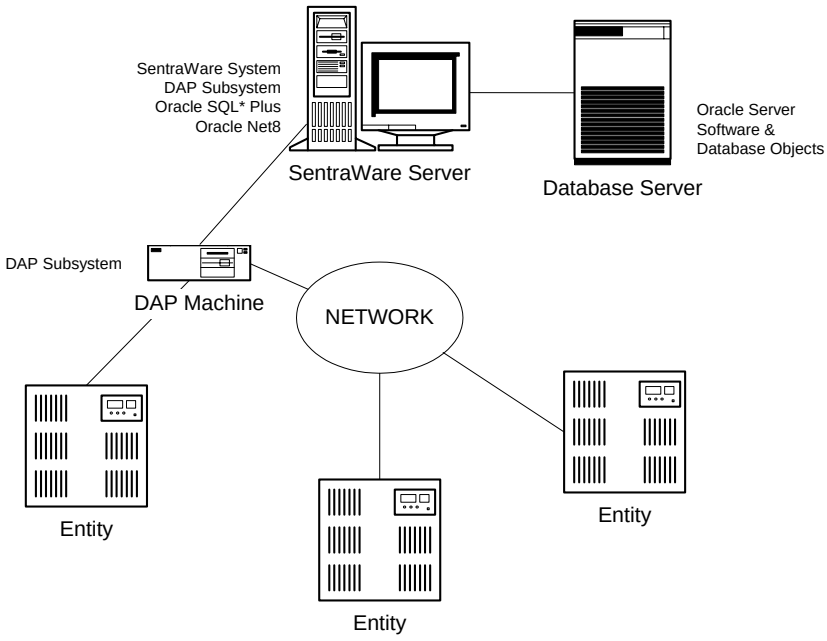


Figure 2.3 - Client/Server/DAP Configuration

- The **DAP** Subsystem software is installed on a separate computer. The **DAP** server provides *SentraWare* with the connectivity to the entities.

The entities may be connected to the **DAP** server over the network or serial lines.

System Requirements

While the design of *SentraWare* supports virtually an unlimited number of entities, the resources required to support a configuration can vary. The following estimates for memory requirements are intended to provide an estimate of the system requirements for 10 DAP default channels, and five *SentraWare* users without GUI applications configuration. Beyond this configuration, required resources are scaled based on the following additional requirements:

- The number of additional users concurrently logged in. Allow an additional 10MB of RAM for each additional user.
- The number of additional **DAP** channels. Allow 1MB of RAM for each additional **DAP** channel running on the **SentraWare** host.

Server-Based Configuration

The following table presents the system requirements for a *SentraWare* server in the Server-Based Configuration.

Item	Requirement
Hardware Platform	Sun Microsystems
Operating System	Solaris 2.6
CPU	Multi-processor
Memory	128 MB (The basic memory requirement is based on five users concurrently logged in to *SentraWare*, and for the default number of DAP channels configured during a typical installation)
Hard Disk	Allow 0.5GB for the operating system, 0.5GB for *SentraWare* files, 0.8GB for Oracle Server products, plus the logging data base and archive files. Use the following formula to estimate the size of the logging data base and archive files: Estimate the number of messages per minute the *SentraWare* host is required to log ("M"). Estimate the number of Entities to be monitored ("E"). (M * 10 * E) / 1024 = Storage requirements, GB/month. Example: Five messages per minute, 50 Entities:(5 * 10 * 50) / 1024 = 2500 / 1024 = 2.5GB per month. Plus 0.5GB for operating system, 0.5GB for *SentraWare*, 0.8GB for Oracle = 4.3GB. Remember this calculation only allows for 1 month of *SentraWare* logging and archiving. Also, this calculation assumes that only message logging or Vterm logging is performed on the entities. If both message logging and Vterm logging are applied on the entities, double the storage requirements.
Data base Software	Oracle8 Server release 8.1.5 distribution kit
SentraWare core	SWAkore package
SentraWare dap core	SWAdpsys package, requires SWAkore
SentraWare engine	SWAeng package, requires Oracle8 Server, SWAkore and SWAdpsys
SentraWare dap devices	SWAdpdev package, requires SWAkore, and SWAdpsys
SentraWare utilities	SWAcli package, requires SWAkore, SWAdpsys and SWAeng

Table 2.1 - System Requirements for a SentraWare Server in the Server-Based Configuration

Client/Server Configuration

The following presents the system requirements for a *SentraWare* server in the Client/Server Configuration.

Item	Requirement
Hardware Platform	Sun Microsystems
Operating System	Solaris 2.6
CPU	Multi-processor
Memory	128 MB (The basic memory requirement is based on five users concurrently logged in to *SentraWare*, and for the default number of **DAP** channels configured during a typical installation).
Hard Disk	Allow 0.5GB for the operating system, 0.5GB for *SentraWare* files, 0.1GB for Oracle Net8 and Oracle SQL*Plus, plus the archive files. Use the following formula to estimate the size of archive files: Estimate the number of messages per minute that the *SentraWare* host will be required to log ("M"). Estimate the number of entities to be monitored ("E"). (M * 5 * E) / 1024 = Storage requirements, GB/month. Example: Five messages per minute, 50 Entities:(5 * 5 * 50) / 1024 = 1250 / 1024 = 1.2GB per month. Plus 0.5GB for operating system, 0.5GB for *SentraWare*, 0.1GB for Oracle Net8 and Oracle SQL*Plus = 2.3GB. Remember, this calculation only allows for one month of *SentraWare* archiving. Also, this calculation assumes that only message logging or Vterm logging is done on the Entities. If both message logging and Vterm logging are applied on the Entities, double the storage requirements.
SQL Software	Oracle Net8 and Oracle SQL*Plus
SentraWare core	SWAkore package
SentraWare dap core	SWAdpsys package, requires SWAkore
SentraWare engine	SWAeng package, requires Oracle Net8 and Oracle SQL*Plus, SWAkore and SWAdpsys
SentraWare dap devices	SWAdpdev package, requires SWAkore, and SWAdpsys
SentraWare utilities	SWAcli package, requires SWAkore, SWAdpsys and SWAeng

Table 2.2 - System Requirements for a SentraWare Server in the Client/Server Configuration

The following resents the system requirements for the database server.

Item	Requirement
Hardware Platform	Any platform that supports Oracle 8
Operating System	Any O/S that supports Oracle8
CPU	Multi-processor
Memory	128 MB
Hard Disk	Allow 0.8GB for Oracle Server products, 0.5GB for the operating system, plus the logging database. Use the following is the formula to estimate the size of logging database: Estimate the number of messages per minute that the *SentraWare* host will be required to log ("M"). Estimate the number of entities to be monitored ("E"). (M * 5 * E) / 1024 = Storage requirements, GB/month. Example: Five messages per minute, 50 entities:(5 * 5 * 50) / 1024 = 1250 / 1024 = 1.2GB per month. Plus 0.5GB for operating system, 0.8GB for Oracle = 2.5GB. Remember this calculation only allows for one month of *SentraWare* logging. Also, this calculation assumes that only message logging or Vterm logging is done on the entities. If both message logging and Vterm logging are applied on the entities, double the storage requirements.
Data base Software	Oracle8 Server release 8.1.5 distribution kit

Table 2.3 - System Requirements for the Database Server.

Client/Server/DAP Configuration

The following presents the system requirements for the **DAP** server.

Item	Requirement
Hardware Platform	Sun Microsystems
Operating System	Solaris 2.6
CPU	Single-processor
Memory	64 MB
Hard Disk	Allow 0.5GB for the operating system, 0.5GB for *SentraWare* files.
SentraWare core	SWAkore package
SentraWare dap core	SWAdpsys package, requires SWAkore
SentraWare dap devices	SWApdev package, requires SWAkore and SWAdpsys

Table 2.4 - System Requirements for the DAP Server.

Installation Session

Solaris Installation Notes

1. When installing Solaris on a *SentraWare* system, choose the full Distribution installation so that all packages required by Oracle are automatically added.
2. When prompted for the disk slicing options, do not accept the default values. Create a separate slice for each option, including but not limited to:
   - /
   - /usr
   - /var
   - /opt
   - /export/home
3. Ensure that at least 150M of disk space is allocated to each slice, and at least 2G is allocated to **/export/home**.

Oracle Installation Notes

1. Log in to UNIX as **root** and perform the following tasks.
2. Configure the Solaris kernel:
   - Edit */etc/system* so that it includes the following defaults. If the values specified in */etc/system* are already set ABOVE these defaults, leave them as is.

     ```
     set shmsys:shminfo_shmmax=4294967295
     set shmsys:shminfo_shmmin=1
     set shmsys:shminfo_shmmni=100
     set shmsys:shminfo_shmseg=10
     set semsys:seminfo_semmns=200
     set semsys:seminfo_semmni=70
     ```

   - Restart the system. Enter the following:
     `shutdown -y -g0 -i6`.
   - Check the file ***/var/adm/messages*** for errors. If the settings in */etc/system* were incorrect, a warning or an error message is presented here.
3. Create 4 mount points on 4 different disks. Distribute the mount points as follows:
   - 1 for Oracle's installation directory
   - 2 for Oracle's database files
   - 1 for *SentraWare*

4. Create the directory */export/home/oracle* using the **mkdir** command.
5. Create a group **dba** using the UNIX command **groupadd dba**.
6. Create a user **oracle** assigned to the group **dba** with the home directory */export/home/oracle*.

   ```
   useradd -d /export/home/oracle -g dba oracle
   passwd oracle
   ```

7. Make a */usr/local/bin* directory, if it doesn't already exist.
8. Log in as **oracle** and perform the following tasks.
   - Place the cdrom in the cdrom drive on the computer where Oracle is being installed.
   - Log on as super user. Enter the following at the UNIX prompt.

     ```
     su -c oracle
     ```

   - Run the **oratab.sh** file.
     - Specify the oracle owner with an environment variable.

       ```
       ORACLE_OWNER=oracle; export ORACLE_OWNER
       ```

     - Run **/cdrom_mount_point/orainst/oratab.sh**.
   - Create oracle's **.profile** file. Place the file in ***/export/home/oracle.*** Include the following in the **.profile** file.

     ```
     ORACLE_HOME=/export/home/oracle/app ; export
     ORACLE_HOME

     LD_LIBRARY_PATH=$ORACLE_HOME/lib:/usr/openwin/lib ;
     export LD_LIBRARY_PATH

     ORACLE_BASE=/export/home/oracle ; export
     ORACLE_BASE

     ORACLE_TERM=vt100 ; export ORACLE_TERM

     PATH=/usr/bin:$ORACLE_HOME/bin:/bin:/usr/ccs/bin ;
     export PATH
     ```

9. Log off, and log back on as **oracle** so that the new **.profile** changes take effect.
10. Run the Oracle Installation. Type the following:

    ```
    cd /cdrom_mount_point/orainst ; ./orainst /m
    ```

11. Select the following options when installing Oracle:
    - Custom Install.
      - Read **preamble.txt**.
      - Read **README.FIRST**
    - Install, Upgrade, or De-Install Software
    - Install New Product - DO NOT CREATE DB OBJECTS
      - Confirm **ORACLE_BASE** & **ORACLE_HOME**

- Confirm Log File Locations
- Confirm Install from CD-ROM
- Confirm National language

- Select the following packages for Installation:
  - Client Software
  - Net8
  - Oracle Intelligent Agent
  - Oracle8 (RDBMS)
  - PL/SQL
  - Solaris Documentation
  - SQL*PLUS
  - JDBC Drivers
- Choose Install
- Confirm DBA Group
- Confirm OSOPER Group
- Answer **No** to the Legato Question
- Select the types of JDBC drivers to install
- Choose Doc Format
- Confirm Installation Completed

12. Log on as **root** and perform the following tasks.
    - Run **$ORACLE_HOME/orainst/root.sh** and input the correct options.

✏ ***NOTE:*** ***The message "Please raise ulimit" is shown REGARDLESS of what the ulimit is currently set at.***

Do NOT update the oracle user's **.profile**. (The Oracle Installation Guide says to update this file; however, it is not required here .)

- Configure Net8, however, there is no need to run net8asst.sh. Make the following changes to the files (only change the responses shown in **bold**):
- Edit */var/opt/oracle/oratab*.

```
#
#  This file is used by ORACLE utilities.  It is created by root.sh
#  and updated by the Oracle8 and SQL*Net install procedures.

#
#  A colon, ':', is used as the field terminator. A new line terminates
#  the entry.  Lines beginning with a pound sign, '#', are comments.
#
#  Entries are of the form:
#      $ORACLE_SID:$ORACLE_HOME:<N|Y>:
#
```

```
#  The first and second fields are the system identifier and home
#  directory of the database respectively.  The third field indicates
#  to the dbstart utility that the database should, "Y", or should not,
#  "N", be brought up at system boot time.
#
#  Multiple entries with the same $ORACLE_SID are not allowed.
#
#
SWA:/export/home/oracle/app:Y
```

- Configure Oracle to startup and shutdown automatically on reboot.
  - Create the **/etc/init.d/dbora** file. The file will resemble the following:

```
# Set ORA_HOME to be equivalent to the ORACLE_HOME
# From which you wish to execute dbstart and dbshut.
# Set ORA_OWNER to the user id of the onwer of the
# Oracle database in ORA_HOME.

ORA_HOME=/export/home/oracle/app
export ORA_HOME

ORA_OWNER=oracle
export ORA_OWNER

if [ ! -f $ORA_HOME/bin/dbstart -o ! -d $ORA_HOME ]
then
echo "Oracle startup: cannot start" >> /dev/console
exit
fi
case "$1" in
'start')
# Start the Oracle databases:
      su - $ORA_OWNER -c $ORA_HOME/bin/dbstart
      su - $ORA_OWNER -c "$ORA_HOME/bin/lsnrctl start listener"
;;'stop')
# Stop the Oracle databases:
      su - $ORA_OWNER -c $ORA_HOME/bin/dbshut
      su - $ORA_OWNER -c "$ORA_HOME/bin/lsnrctl stop listener"
 ;;
esac
```

  - Link the following files to the appropriate startup/shutdown files:

```
ln -s /etc/init.d/dbora /etc/rc2.d/S94dbora
ln -s /etc/init.d/dbora /etc/rc0.d/K21dbora
```

SentraWare Installation Notes

1. Log on as **root**, and mount the installation medium where the *SentraWare* packages are located. For example, enter:

   `mount -F nfs <hostname>:/tone/swadev/pkg /mnt` **ENTER.**

2. Add the ***SWAkore*** package using the **pkgadd** command.

- Specify the user and group to be created. It is recommend that the defaults of **swa** (user) and **swa** (group) are selected.

3. Log on as **oracle** and perform the following tasks.
   - Run **$SWATOP/etc/conf/oracle/makedb**. ($SWATOP is the directory specified as the *SentraWare* home directory during the installation of SWAkore.)
     - Specify **SWA** for the SID.

       The same mount point may be specified for both prompts if there is only one disk, or two of the four mount points can be specified if they were created in Step 3 of the Oracle Installation.

       If the defaults are not accepted while running **makedb**, you must repeat most of the steps already covered, and change the information accordingly. It is strongly suggest that the defaults are accepted.
4. Start the database, manually. Enter the following at the UNIX prompt:
   ```
   $ORACLE_HOME/bin/dbstart
   ```
5. Start the listener, manually. Enter the following at the UNIX prompt:
   ```
   lsnrctl start
   lsnrctl status
   ```
6. Check Oracle Database connectivity:
   - Run ***sqlplus***
   - Log on as **system/manager**
   - If an **SQL>** prompt is received, the database is running. Type:
     `exit` **ENTER**.
   - Check that Net8 is working:
     Type:
     `sqlplus swa/alert@swa.hostname` **ENTER**.
     (as defined in tnsnames.ora)
   - If an SQL> prompt is received, then Net8 is working. Type `exit`.
7. Log in as **root** to perform the following tasks.
8. Add the ***SWAdpsys*** package using the **pkgadd** command.
9. Add the ***SWAeng*** package using the **pkgadd** command.
   - The Oracle database user name is **oracle**

- The database server is **swa.hostname**, (as configured in tnsnames.ora, step 17)
- The database user name is **swa**
- The database user password is **alert**
- Initialize the database. (Enter `yes` at the prompt.)

10. Add the ***SWAdpdev*** package using the **pkgadd** command.
11. Add the ***SWAcli*** package using the **pkgadd** command.

Post-Installation Tasks

This section describes tasks that must be performed after completing the *SentraWare* installation.

- Verifying the Installation
- Complete System Backup
- Administer *SentraWare*

Verifying the *SentraWare* Installation

The following procedure is not a comprehensive troubleshooting guide, but rather a walkthrough of the installed *SentraWare* system with its related database tools. This procedure is also used to acquaint the new administrators and users with some of *SentraWare*'s functions.

At the UNIX prompt, enter **swash** and press **ENTER**. UNIX prompts for the login and password.

```
$ swash

swash login: sadm
password:
sadm @
```

Login as **sadm** user and perform the following tasks while at the **swa** prompt (@):

1. List the available *SentraWare* commands.
2. Run the verify commands. (See "Run the Verify Commands", below.)
3. Run **op-stat-ent** and **restore-log** commands.
4. Run browse command.
5. Terminate and restart *SentraWare*.

List the Available *SentraWare* Commands

At the **swash** prompt (@), type **?** to display the available *SentraWare* commands.

```
@ ?

Valid commands are:
access        filter            rc-cent      verify-act
allow-log     inhibit-log       rc-ent       verify-cent
allow-system  inhibit-system    rc-sched     verify-ent
archive       output-stat-ent   rc-usr       verify-perm
bin           output-system     restore-log  verify-sched
browse        playback          remove-log   verify-usr
swash         rc-act            verify-access

@
```

*Figure 2.4 - SentraWare **swash** Commands*

Run the Verify Commands

1. At the **swash** prompt (@), enter the **verify-cent** command. Validate the test center is listed.

   The **test** center is the predefined center in *SentraWare*.

2. At the **swash** prompt (@), enter the **verify-ent** command. Validate the **myself** entity is listed.

   The **myself** entity is the machine where *SentraWare* is running. This entity allows *SentraWare* to report potential problems within the system.

3. At the **swash** prompt (@), enter the **verify-usr** command. Validate the **sadm** user is listed.

Run op-stat-ent and restore-log

1. Enter the **op-stat-ent** command. Validate the "Entity Status Report" is displayed, to list availability and logging status of the monitored entities.
2. Enter the **restore-log** command. Respond with **?** at the **entity=** prompt to display the help text along with the list of defined entity names.

```
@ restore-log

entity = ?

Enter a single entity name as defined with 'rc-ent'.

Valid entity names are:
myself   telnet1   test1   test2   test3
```

3. Enter **myself** at the **entity=** prompt. One of the two scenarios below are shown:

```
PF

Logging already on for entity myself
  OK
```

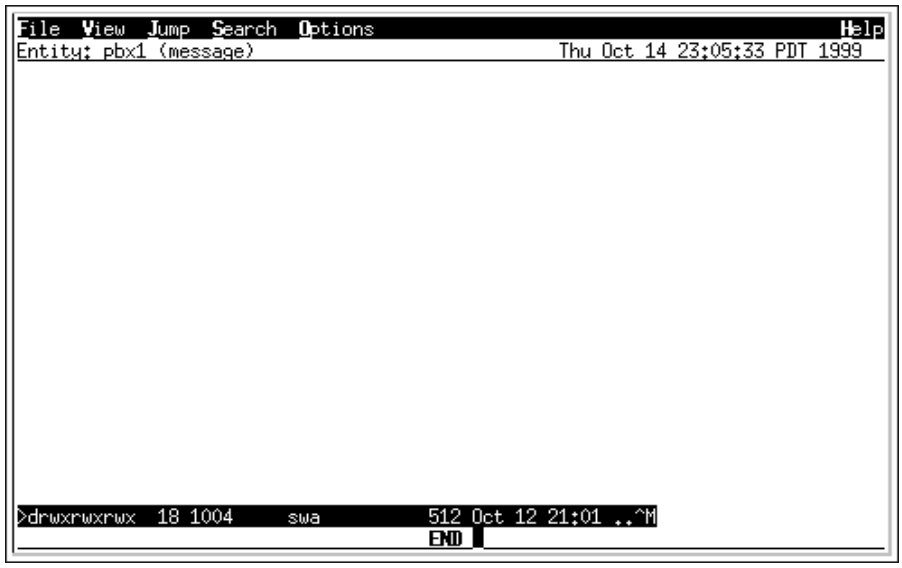
or

```
PF
  OK
```

Run the browse Command

1. The *SentraWare* Log Browser allows you to view the contents on an entity's log file. Test the Log Browser by entering **browse** at the **swash** prompt (**@**) and select **File** (**SHIFT**+**F**), **Open Log**, **myself**.

   A screen similar to the following is displayed.



   "102:112 Available" should be logged in the message log data base every time an Entity logging is restored.

Terminate and Restart *SentraWare*

1. At the **swash** prompt (@), enter the **output-system** command. Validate that the "System Status Report" is displayed.
2. Enter the **inhibit-system** command. Validate that *SentraWare* terminates.
3. Enter the **allow-system** command. Validate that *SentraWare* restarts.

Complete System Backup

Perform a complete system backup upon a successful *SentraWare* installation.

See your UNIX System Administrator's Guide for the operating system commands **ufsdump**, **ufsrestore**, and **tar** to ensure the proper backup of the operating system.

Administer *SentraWare*

The *SentraWare* Initialization File

The *SentraWare* initialization file **SentraWare.ini** is located in the **swa/etc/** sub-directory. The following is a sample of the contents of **SentraWare.ini** file.

```
[Primary Database]
Database = swa.sun
User     = swa
Password = 001f01420109160617
```

The password is encrypted. To calculate the password, use the **dbpassword** utility located in the **SWATOP/etc/conf/bin** sub-directory.

```
$ dbpassword testing+
17151f46454e
$
```

In the example above, how **dbpassword** calculates the password testing+ is shown.

Stop and restart *SentraWare* after completing the changes to the initialization file. Use the following steps to stop/restart *SentraWare*:

1. Login as super-user
2. From the UNIX prompt, type the following to stop *SentraWare*:

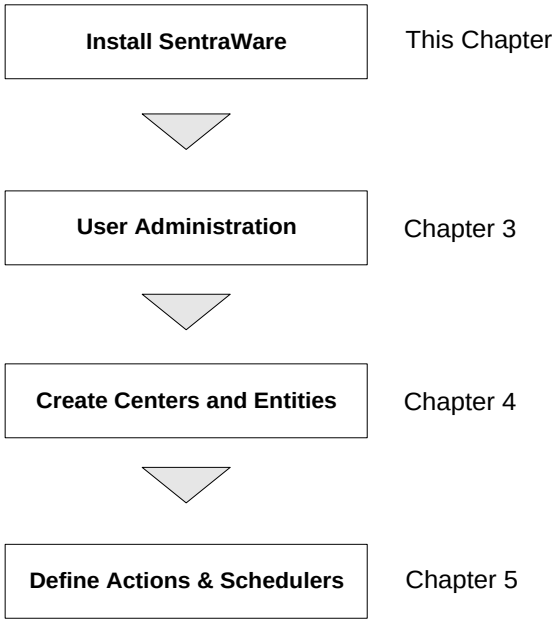
```
# /etc/init.d/sentraware stop
```

3. To restart *SentraWare*, type the following:

```
# /etc/init.d/sentraware start
```

Administer New *SentraWare* Installation

The following figure illustrates where to find information about the new *SentraWare* procedure.



Chapter 3 - User Administration

Overview

This chapter describes the various types of *SentraWare* users and administrators and how user permissions are managed.

There are four (4) major *SentraWare* functions:

- Administrative
- Browse
- Access
- Operate

Each function class has restrictions relative to the use of the system. For example, an Operate user cannot enter a command that changes a system-wide parameter.

Some of the activities performed by organization personnel may overlap. This means that one user may be responsible for more than one function. Also, allocating system activities among personnel may vary between installations. Each user classification can be thought of as a functional set of activities and not just a job description.

There is no limit to the number of *SentraWare* users.

Topics to be discussed

- **System Administrator,** describes the tasks of a System Administrator and the procedure to create a System Administrator.
- **Maintaining SentraWare Users,** describes the procedure to create a Center Administrator, other *SentraWare* users, and verification.
- **Managing User Permissions**, describes the procedures to assign permissions to *SentraWare* users.

Commands used to perform the administrative tasks listed above:

Command	Application
swash	Enters the *SentraWare* Command Language environment.
rc-usr	Creates, modifies, and deletes users on the *SentraWare* system.
verify-usr	Outputs a list of valid user login identifications for all *SentraWare* users.
verify-perm	Outputs a Permission Verification report. The user(s) name(s), permission(s), and the assigned center(s) are displayed.
verify-cent	Displays all center names defined in the system.
rc-cent	Used to add, change, or delete a center. After the center has been defined, entities are assigned to the center using the **rc-ent** command.

NOTE: ***Consult the SentraWare Reference Manual for complete descriptions of the commands discussed in this section.***

System Administrator

The System Administrator is the *SentraWare* user who is responsible for the overall operation of *SentraWare*. In the current release of *SentraWare*, the System Administrator can be identified by any login name. The System Administrator must have **ADMIN** permission. A System Administrator may create Center Administrators. All the commands used by the Center Administrators can be executed by the System Administrator. A System Administrator who does not create a Center Administrators is responsible for administering ALL *SentraWare* operations.

The following table describes the System Administration functions by task type.

The System Administrator **creates**…	Logins for *SentraWare* users.
	Automatic response scripts.
	Centers, and allocates system resources to centers.
	Maintains a master list of all entities, users, patterns, and user programs.
The System Administrator **establishes**…	A schedule for the generating system reports.
	The system backup (data base and logging files) and maintenance schedule.
	User programming guidelines.
	Special access permissions.
	SentraWare parameters.

Table 3.7 - Functions of the SentraWare System Administrator. (Part 1 of 2)

The System Administrator **changes**…	Passwords for system-delivered logins, such as **root**.
The System Administrator **assigns**…	User-programming permissions to individual users. Devices across center boundaries.
The System Administrator…	Periodically **checks** the status of *SentraWare* users. **Monitors** Scheduler execution. **Performs** *SentraWare* audits to maintain system integrity.

Table 3.1 - Functions of the SentraWare System Administrator. (Part 2 of 2)

Initially, the **root** user creates a user name **sadm** and assigns **sadm** a password by using the *SentraWare* **rc-usr** command. Once a password is assigned, **sadm** assigns user names and passwords to all *SentraWare* users.

NOTE: In the current release of SentraWare the rc-usr command does not create the UNIX user. The rc-usr command only creates a SentraWare user. To add a UNIX user, use "adduser" (NCR UNIX MPRAS) or "useradd" (Sun Solaris). Refer to the UNIX System Administrator Manual for further details about adding a UNIX user. The rc-usr assigns SentraWare user name, password, and the permissions.

To create a System Administrator using **sadm** for the first time:

1. Execute **swash**, respond to the swash login prompt with **sadm** and the password prompt with the supplied password.
2. Execute the **rc-usr** command to create, change, or delete *SentraWare* users.
3. Respond with **n** at the **rctype** prompt. The command prompts for the new user name(s) to be created. The *SentraWare* user name does not have to be the same as the UNIX login name. Enter a text string of fifteen or fewer characters (with no leading or trailing blanks), beginning with an alphabetic character. All other characters in the name must be alphabetic, numeric, or a hyphen (-). To simplify system administration, a user's last name can be used as the user name.
4. The **rc-usr** command also prompts for the following information:
   - The **password** to be assigned to the user (entered twice to validate). Respond to this prompt with a password of fifteen or fewer characters (with no leading or trailing blanks).

NOTE: If a slash (/) is used, the result is the slash terminates this command and does not becomes part of the password.

- **Equivalent users and hosts**

  Respond to this prompt with the UNIX login name. Enter equivalent users and hosts using the form

  ```
  <Unixuser>@<hostname> , ...
  ```

  if @<hostname> is omitted, a UNIX user from any host may log in.

NOTE: ***If the UNIX login name is used in the equivalent user field, after logging into UNIX, it is not necessary to specify the SentraWare user name and password to enter the swash (SentraWare Command Language) environment. This enables execution of all of the SentraWare commands under the UNIX shell.***

- **Info**. A comment field for additional information.

  Anything may be entered in this field. Enter the user's full name, phone number, and room number.

5. Verify the user name just created using **verify-usr** command. If the results of the verification is satisfactory, logoff as **sadm** and login with the name just created. For more information about the **verify-usr** command, refer to the *SentraWare Reference Manual*.
6. Execute **rc-usr** with **rctype = o** to delete the user **sadm**, or **rctype = c** to change the password of **sadm**. The system is now secure.

Maintaining *SentraWare* Users

Creating Center Administrators (cadm)

Before creating a Center Administrator, create a *SentraWare* center using the **rc-cent** command.

Execute the **verify-cent** command to display the parameters available for existing centers.

To create a Center Administrator:

1. Execute the **rc-usr** command.

   The command prompts for the user name to be created.

2. Respond with **n** at the **rctype** prompt.

   The command prompts for the new user name. The *SentraWare* user name does not have to be the same as the UNIX login name. Enter a text string of fifteen or fewer characters (with no leading or trailing blanks), starting with an alphabetic character. All other characters in the name must be alphabetic, numeric, or a hyphen (-). To simplify system administration, a user's last name can be used as the user name.

3. The **rc-usr** command also prompts for the following information:
   - The **password** to be assigned to the user (entered twice to validate). Respond to this prompt with a password of fifteen or fewer characters (with no leading or trailing blanks).

NOTE: ***If using a slash (/), the slash terminates this command. The slash does not becomes part of the password. Later, user's may change the assigned password using rc-usr.***

- **Equivalent users and hosts**

  Respond to this prompt with the UNIX login name. Enter equivalent users and hosts using the form

  ```
  <Unixuser>@<hostname> , ...
  ```

  if @<hostname> is omitted, a UNIX user from any host may log in.

NOTE: ***For security, do not place a UNIX user in the equivalent list. When executing swash, this user is prompted for the SentraWare user name and password.***

- **Info**. A comment field for additional information.

  Anything may be entered in this field. Enter the user's full name, phone number, room number, etc.

Verifying Users

Use the **verify-usr** command to check the information associated with the login. Information fields available are:

Fields	Application
User Name	The user login.
Equiv. User	The equivalent UNIX login name.
Equiv. Host	The host name associated with Equiv. User.
Info	The user information.

*Table 3.2 - Information Fields Available with **verify-usr** Command*

The command format is:

```
@verify-usr user=<username> <t>
```

A sample of the **verify-usr** command and the resulting output is displayed below:

```
@ verify-usr user=jad
  PF


                    User Verification Report
                    Fri Aug 20 17:53:03 1999

         USER       EQUIV   EQUIV
         NAME       USER    HOST            INFO
         ----       -----   -----           ----
         jad        jad     puma        John Doe

@
```

*Figure 3.1 - Sample **verify-usr** Command and Output*

Verifying Permissions

Use the **verify-perm** command to check the permissions associated with the login. Information fields available are:

Fields	Application
User Name	The user login.
Perm	The permission(s).
Center	The center name associated with the permission.

*Table 3.3 - Information Fields Available with **verify-perm** Command*

The command format is:

```
@verify-perm user=<username> <t>
```

The following example displays the **vfy-perm** command and the resulting output

```
@ verify-perm
  PF


             Permission Verification Report
               Fri Aug 20 15:10:08 1999

USER
NAME             PERM            CENTER
----             ----            ------
adh              OPERATE         <any>

budi             ADMIN           <any>
                 ACCESS          <any>
                 BROWSE          <any>
                 OPERATE         <any>

jad              <none>
```

Figure 3.2 - Permission Verification Report

Creating Other Users

The System Administrator is responsible for assigning, changing or deleting logins, passwords, and permissions for *SentraWare* users, where appropriate.

To add users, execute the **rc-usr** command as described previously.

- Assign the logins, passwords, equivalent users, and permissions.

Managing User Permissions

The functions performed by a user in *SentraWare* are controlled by the set of permissions granted by the System Administrator. These special privileges/permissions allow a user to administer entities in several centers, to control the logging process of entities in several centers and to access the UNIX system and its programming facilities to effect entity automation.

The System Administrator uses the **rc-perm** command to create, modify, or delete *SentraWare* user permissions. This command prompts for the **permission** information. A *SentraWare* user has NO PERMISSIONS by default.

The flexible user permission feature in *SentraWare* allows the System Administrator to assign the required permissions to individuals according to their responsibilities and their assigned resources.

Assigning ACCESS Permissions for Access Function

SentraWare users can enter access mode with entities in their own center using the **access** command.

The System Administrator uses the **rc-perm** command to create, modify, or delete an access permission for a user. This command prompts for the **permission** information. The following example demonstrates how to create an ACCESS permission for the user "**trial**", and for entity "**localhost**".

```
@rc-perm/
rctype = new/
user = trial/
permissions = ACCESS@localhost/
@
```

Figure 3.3 - Creating an ACCESS Permission for a User

To display the permissions for a user, execute the **verify-perm** command. For more information about the **verify-perm** command, refer to "Verifying Permissions" section on page 3.8 of this chapter or the *SentraWare Reference Manual*.

Assigning BROWSE Privileges for Browse, Filter, and Playback Functions

SentraWare users can view entity logging data through the **browse**, **filter**, and **playback** commands. (For details, see the *SentraWare Reference Manual*.)

NOTE: ***A user who has no ACCESS permission but has BROWSE permission, may have access in READ-ONLY mode. In READ-ONLY mode, the user cannot input information from the keyboard.***

To create, modify, or delete a special **browse** permission for a user, the System Administrator uses the **rc-perm** command, described in "Assigning ACCESS Permissions for the Access Function". The following example demonstrates creating a **browse** permission for the user named "**trial**" so that the user can browse all entities.

```
@rc-perm/
rctype = new/
user = trial/
permissions = BROWSE
@
```

Figure 3.4 - Creating a BROWSE Permission for a User

Assign OPERATE Permissions

SentraWare provides allows the user to monitor operations, track alarms, and take specific actions according to their center's procedures. The **OPERATE** permission provides authorization to modify dynamic *SentraWare* configurations (remove entity log, inhibit polling, message alerting, alarms, etc.).

To assign **OPERATE** permissions to users, the System Administrator uses the **rc-perm** command. The following example demonstrates creating the OPERATE permission on the "**test**" entity for the user named "**trial**".

```
@rc-perm/
rctype = new/
user = trial/
permissions = OPERATE@test
@
```

Figure 3.5 - Assigning OPERATE Permissions

Multiple permissions may be created for a user. The following example shows the ALARM, OPERATE, and BROWSE permissions are assigned to the user named "trial" on the entity "**test**":

```
@rc-perm/
rctype = new/
user = trial/
permissions = ALARM, OPERATE, BROWSE@test
@
```

In change mode, typing **?** at the **permissions=** prompt displays the available permissions list and the list of permissions assigned to the user. For example:

```
@rc-perm/
rctype = c/
user = trial/
permissions = ?

   Enter permissions for this user in form
       <permname>@<center>

   if @<center> is omitted, user will have this
   permission for all centers.

   Valid permissions are:

         ACCESS              - Access entities
         ADMIN               - Changing entity name, center and user admin
         ALARM               - Acknowledge/clear alarms
         BROWSE              - Browse entity logs
         OPERATE             - Inhibit/allow commands

    Current permissions are:

          ACCESS
          BROWSE
          OPERATE
permissions=
```

List Users Currently Logged On

The executable command **bin who** lists the users currently logged on to *SentraWare*. Information provided by this command are the user identification, the line number, and the time that the user logged in. An example of the **bin who** command and the system response is shown in the following figure:

```
@ bin who
 PF
command->(who)
mph         pts/16       Dec 10 09:31    (mph.tonesoft.com)
jad         pts/4        Dec  9 11:54    (moose.tonesoft.com)
sadm        pts/5        Dec  9 11:53    (xuga )
jad         pts/7        Dec  9 11:56    (moose.tonesoft.com)
jad         pts/8        Dec  9 12:06    (moose.tonesoft.com)
mph         pts/33       Dec 14 17:16    (mph.tonesoft.com)
bystr       pts/45       Dec 14 11:58    (moose.tonesoft.com)

@
```

*Figure 3.6 - Sample **bin who** Listing*

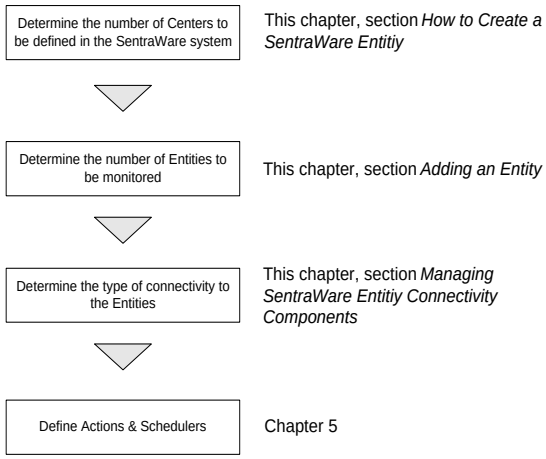
Chapter 4 - Managing the System

This chapter contains information about setting up *SentraWare* and the routine maintenance required to keep it functional (for example, archiving, connectivity, troubleshooting).

NOTE: ***To perform System Administration tasks described in this chapter, requires ADMIN .***

Building the System

The following figure illustrates the steps used to build a *SentraWare* system and reference the section of the document that explains the steps:



Creating a *SentraWare* Entity

Before an entity is created, a center must be defined. Centers are a logical grouping of entities most commonly based on geographic location, department, or entity type. Used to administer *SentraWare*, each center can be thought of as a virtual *SentraWare*.

Centers are defined and assigned to a center using the **rc-cent** command. Once a center has been defined, it is complex to change because underlying databases are based on the center definition. Only the System Administrator can assign users to a center.

The systems in a center need not be located in the same geographic area, since center boundaries are administrative rather than physical. In the current version of *SentraWare*, there is no limitation to the number of centers.

✏ ***NOTE:*** ***The System Administrator establishes the center and the Center Administrator login.***

1. Execute the **rc-cent** command from the *SentraWare* system prompt to create a center.
2. Enter `rc-cent` and press **ENTER.**

   The **rc-cent** command prompts for the type of change.

3. Enter `n` and press **ENTER.** (The forward slash (/) may be used to accept the default values and display the next prompt.)

   The **rc-cent** command prompts for the name of the center.

4. Enter the center name and press **ENTER.**

   The center name can be any alphanumeric string of up to fifteen characters, starting with an alphabetic character. The center name is case sensitive. If an attempt is made to remove a center with existing ports or users (including the Center Administrator) assigned to it, *SentraWare* outputs an error message. The user is stopped from removing the center.

The example on the next page demonstrates how to create a center called **acomps**. Also shown is an example of using the **verify-cent** command to list all centers defined in *SentraWare*.

```
@ rc-cent

rctype = n/

center = acomps/
  PF
  OK


@ verify-cent
  PF


                Center Verification Report
                Mon Dec 28 16:15:00 1998

CENTER NAME
-----------
acomps
test
```

*Figure 4.1 - Example of Creating a Center Using **rc-cent** and **verify-cent***

Adding an Entity

Use the **rc-ent** command to assign the entities to the centers. The **rc-ent** command includes information about the following items:

- Entity name
- Center name associated with the entity
- Channel name associated with the entity
- Scheduler names to be used
- Message Alerting table names
- Number of days to store the entity log in logging database.

There is no limit to the number of entities that can be defined with the **rc-ent** command.

The **rc-ent** command is restricted to the System Administrator. When creating an entity, the Administrator responds to a series of prompts as presented in the example below:

```
@ rc-ent

rctype = n/
entity = a-sparrow/
center = acomps/
channel = //eagle/tty01/
logtime = 60/
logpattern=
timeout=
mapattern=newton
schedule = /
table = s-siska/
info = "NCR Server in Building A"/  PF
  OK

@
```

*Figure 4.2 - Example of Adding an Entity (**rc-ent** Command)*

Entering a forward slash (/), selects the default value for each parameter. The value may be redefined later. Entering a question mark (**?**) at a prompt, displays help text that provides instructions and applicable selection values, including previously entered values.

The **channel**, is the name of the channel used to connect the entity to the system. The channel name does not have to exist. In the example above, the Channel Name is **tty01**. The Channel Name is defined in the host **eagle**. Specify only the Channel Name and omit the hostname field if the DAP subsystem is located in the same machine where *SentraWare* is running. It is not necessary to specify the connectivity details at this point. The information about how to define a channel is described in the next section, “Managing SentraWare Entity Connectivity Components”.

Other parameters, such as the **center**, **schedule** and **table** must exist. (See the *SentraWare Reference Manual* for a full explanation of the **rc-ent** command. See Chapter 5, “Working with Actions and the Scheduler” of this document for information about creating **actions** and **schedulers**). If **schedule** or **table** has not been defined, enter **“none”** or a forward slash (/) at the appropriate prompts.

The following table defines the prompts encountered when creating an entity:

rctype	Respond with the type of the recent change to be made. The valid responses are: **new** To add a new entity in the database **chg** To modify an existing entity in the database **out** To delete an existing entity in the database (These responses may be abbreviated **n**, **c**, and **o**, respectively.) Default response: There is no default value.
entity	For rctype **new**, the name must not exist and must be no more than 15 alphanumeric characters (including a hyphen (-)) long. The name is case sensitive and must begin with an alphabetic character. A colon cannot be used. The name entered must not be the name of an existing entity in the database. Care must be exercised in choosing an entity name; once it is assigned it may not be changed by using **rctype = chg**. If rctype is **chg** or **out**, the entity must exist. The user must have ADMIN in the named entity. If rctype is **out**, the command deletes all the named entity's entries in the database. The associated information defined in the entity database is also removed. No further prompting is made. Default response: There is no default value.
center	This prompt is issued if the rctype is **new** or **chg**. The user must have ADMIN permission in the named center. If rctype is **new**, enter an existing center name for the entity being created. If rctype is **chg**, enter an existing center name for where the entity is associated. Default response: There is no default value if rctype is **new**, or the current value if rctype is **chg**.
channel	Enter channel name for the entity in form of an URL: `[protocol:][//hostname[:port]/]channelname` The channel name must match the name configured in the DAP subsystem. The port number is defined in the SWATOP/etc/dapmgr.ini. If not specified, the default value is **8003**. Enter an IP address or a host name in the hostname field. The default value for hostname is the local host. Default response: The default is **none** if rctype is **new**, or the current value if rctype is **chg**.
logtime	Enter the number of days to store the entity log in the logging database. The default is **30** days if rctype is **new**, or the current value if rctype is **chg**.

*Table 4.1 - **rc-ent** Command Prompts (Part 1 of 2)*

timeout	Specify the maximum time, in seconds <sec>, the entity has to start a module until it fails and generates an error message Default tolerance is **120** if the rctype is **new**, or the current value if rctype is **chg**.
mapattern	Specify the MA Pattern the entity is to use.
schedule	Specify names of the schedules assigned to the entity. The schedule names entered must be existing schedule names, created using **rc-sched**. Specify **none** to clear the current list. Responding with **?** list the schedules currently assigned. The default is **none** if rctype is **new**, or the current value if rctype is **chg**.
table	Enter the names of the MA Tables assigned to the entity. The MA Table name must be no longer than 15 characters. Specify none to clear the current list. Responding with **?** list the table currently assigned. The default is none if rctype is **new**, or the current value if the rctype is **chg**.
info	Enter additional information about an entity specified. Enclose the information in double quotes (“). The default is **none** if the rctype is **new**, or the current value if the rctype is **chg**.

*Table 4.1 - **rc-ent** Command Prompts (Part 2 of 2)*

Use the **verify-ent** command to view parameters entered for an entity.

Managing Entity Connectivity Components

The Data Acquisition Subsystem performs all the data collection management and response delivery functions for *SentraWare*. *SentraWare* doesn't communicate directly with an entity; it uses the Data Acquisition Process Server (**DAP**).

The **DAP** receives data from monitored entities, performs the necessary preprocessing (using specified Terminal Emulation), and delivers the data to the data logging mechanism. It also takes data from the response mechanisms of *SentraWare* and delivers these to the entities and contact closure relays.

Between the DAP and *SentraWare* are other software modules, the **Channel Manager** and **Data Dispatcher**.

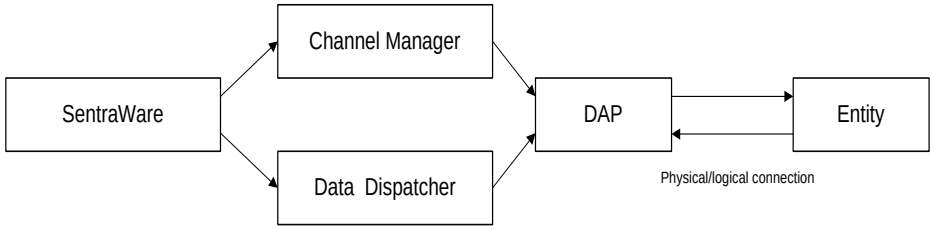


Figure 4.3 - Data Acquisition Subsystem Flow

The term **DAP** refers to:

- Data Acquisition Process
- **DAP** Server

The **DAP** Server is the physical host machine where the Data Acquisition Process resides. It can be the *SentraWare* host machine, or a separate UNIX machine located on the network (Remote DAP Server). A remote **DAP** Server may be in a different room, a different state, or even a different country, executing one or more **DAP** processes. The **DAP** Server acts as the connection point between *SentraWare* and its monitored entities.

The **DAP** is the executable binary file (like a DOS driver device) that receives data from the monitored entities, performs the necessary preprocessing (using specified Terminal Emulation) and delivers the data to *SentraWare* using TCP. All responses from *SentraWare* to the entities, such as input messages and actions, also pass through the **DAP**.

SentraWare monitors/provides access to an entity by connecting a channel on the entity to a channel on the **DAP**. (Any security issues are handled at the entity level.)

The **DAP**s are defined to carry out instructions independent of *SentraWare*. The location of a **DAP** process is transparent to *SentraWare*.

The following design concepts are implemented in the Data Acquisition Subsystem:

- **Client/Server technology**. The subsystem is the server and the *SentraWare* system(s) are the clients that request services specified by a protocol to the Data Acquisition Process.

- **Distributed parallel processing**. Data processing takes place in each Data Acquisition Process (**DAP**), therefore, every logical line has its own programmable data collection and response delivery mechanism.
- **Network connectivity**. All connections between components are made over a TCP/IP network, therefore, components can be placed on different machines across the network.
- **UNIX connectivity**. Together, with serial and telnet connection **DAP**s, there are **DAP**s that provide generic connectivity to any UNIX processes using the standard **ptx** mechanism. **DAP**s can be programmed to connect to additional filtering and connectivity agents (DAP Application, Channel Processor). Any UNIX application can be specified as a **DAP** application in terminal emulation mode, therefore, the application interacts with a *SentraWare* I/O line as if it were a terminal.

The Data Acquisition Subsystem consists of the following parts:

- **Channel Manager**. Coordinates outgoing message traffic sent to an entity. **Channel Manager** is the **chanmgr** process on the *SentraWare* machine. **Channel Manager** accepts message traffic for a channel and notifies software modules about the status of their requests.
- **Data Dispatcher**. Receives filtered data through the logical lines and delivers it to the Comparator, VT Logger, or Message Analyzer, regardless the nature of the data. **Data Dispatcher** permanently connects to the local **Event Dispatcher** and uses shared memory to deliver an input stream from the **Event Dispatcher** internal data buffer to the logger.
- **Data Acquisition Point Processors (DAPs)**. Connects to entities and the **Channel Manager** and **Data Dispatcher**. They perform all the filtering of the incoming data and the transmission of the *SentraWare* responses. Basically, **DAP**s serve as remotely controlled hardware drivers with built-in filtering features. They maintain settings of every I/O line and execute terminal emulation modules to process entity input data. **DAP** processors may run on the same UNIX computer as *SentraWare*; or can be distributed over a network, and can be placed on dedicated servers (**DAP** Servers); or on any shared UNIX machine where data collection and preprocessing is required. **DAP** servers perform the roles of intelligent and fault-tolerant terminal servers, providing the most direct and robust way to connect entities to *SentraWare*.

In the current release of *SentraWare*, a **DAP** can be connected to more than one *SentraWare* system. This feature adds fault resilience to the system since a monitored entity can be administered by more than one *SentraWare* system. Also the **DAP** can take a screen snap shot of the monitored entity. The snapshot is useful during **access** and **playback**. The user can playback the logging file and examine the screen image, see Chapter 8 of the *SentraWare Reference Manual* for more information, of the monitored entity console in full screen mode.

- **DAP Manager (dapmgr)**. **dapmgr**s dynamically serve the requests from any Channel Manager on any host and issues necessary commands to perform required functions. **dapmgr** dynamically starts a **DAP** process upon receiving a request from a **Channel Manager** on any host to open a defined channel. If the channel is closed (as the result of **remove-log** command), **dapmgr** stops the DAP process associated with the channel and releases the machine resources that were used by the process.

All components of the Data Acquisition Subsystem have built-in fault resilient mechanisms to provide unattended, rapid, automatic recovery from processing outages experienced by the system. For example, turning the power of a **DAP** machine off and on results in one to three minutes of data loss due to the reboot process. After the reboot, all settings on all I/O lines are completely restored and **DAP** operation continues. The self-monitoring and tracing mechanisms allow early detection of a failure (for example, carrier failure or loss of **DAP** heartbeat). Through the *SentraWare* alarming mechanisms, these conditions are recognized, automatic processes invoked, and the operations staff alerted (if manual intervention is required).

At *SentraWare* startup, **DAP Manager**, **Channel Manager,** and **Data Dispatcher** are started by *SentraWare*'s **Executor** program (see “Controlling the SentraWare Processes” on page 4.16 of this chapter). Connections to perform I/O among these and other processes like **Poller**, **Comparator**, **VT Logger,** and **Message Analyzer** are established dynamically, as needed.

Setting Up a DAP

When preparing to connect an entity to *SentraWare,* note the following points:

- What type of system is being monitored?
- What type of connection will be used?

The following types of connections can be used to connect *SentraWare* to an entity:

- A **direct serial** interface. - For example, an RS-232 cable.
- A **serial over TCP** interface. - A serial connection from the entity to a **terminal server**. The terminal server, in turn, is connected to *SentraWare* by a network connection.
- A **direct network** interface. - *SentraWare* and the entity are on the same Ethernet or Token Ring network. The following possibilities then emerge:
  - Telnet; if TCP/IP is enabled.
  - A remote shell, script or proprietary network program.

The following connection aids may be required:

- **Serial multiplexors/cards**. Communicates with the *SentraWare* host. Serial multiplexors/cards allow connection of many devices, serially, to *SentraWare* or a remote **DAP** server. They are used when the number of serial ports is limited.
- **T-Tap connectors**. Splits the signal and sends a copy to another system.
- **Protocol converters**. (For example, a DynaStar.) Converts one type of input signal to a (different) output signal. It can convert input X.25 data to TCP.
- **Modems**. For dial-up entities.

Specific **DAP**s are provided for specific types of connections. The following **DAP**s are available with additional **DAP**s to be supplied in the future. (Refer to the *SentraWare DAPs Configuration Guide* for more information.)

DAP	Application
dapsyslog	Built-in system console log support. The **DAP** monitors performance characteristics of the computer where the **dapmgr** is installed.
dapexec	**DAP Application**, supports CHAPs.
dapfile	**DAP** to monitor a file (similar to dapexec with "tail -f <filename>" as the CHAP).
dapserial	Supports a connection using serial asynchronous port.

The **dapsyslog** is provided with the *SentraWare* system, while the other **DAP**s are add-ons.

The **DAP** executable programs are present in the **SWATOP/sbin** directory.

The **DAP** subsystem for *SentraWare* has been designed to make the configuration process straightforward. The channel configuration can be performed by editing the **DAP** configuration files. (Refer to the *SentraWare DAPs Configuration Guide* for information about building **DAPs**.)

DAP Configuration Files

There are two **DAP** configuration files:

- **ctype.conf**
- **cdata.conf**

The files are stored in the **SWATOP/etc/conf** sub-directory.

Using ctype.conf to Define the Types of DAP

The file **ctype.conf** defines the maximum number of channels and every **DAP** type. The following table describes the keywords used in **ctype.conf**.

Keyword	Description
maxChan	Specify the maximum number of channels.
[LABEL]	The name of the **DAP** channel type (**chanType**).
progName	The **DAP** program name for the specified **DAP** channel type. The **DAP** program name must be installed in the **sbin** sub-directory.
formName	The name of the HTML file used for the data entry panel. The HTML file guides the user during channel configuration process using Java Enabled Internet Browser.

Table 4.2 - ctype.conf Keywords

The following is an example of a **ctype.conf** file:

```
maxChan = 10

[SYSLOG]
progName = sbin/dapsyslog

[FILE]
progName = sbin/dapfile
formName = dapFile.html

[SERIAL]
progName = sbin/dapserial
formName = dapSerial.html

[APPLICATION]
progName = sbin/dapexec
formName = dapSerial.html
```

Figure 4.4 - Sample ctype.conf

In the example above, there are four DAP **chanTypes** defined: SYSLOG, FILE, SERIAL, and APPLICATION with the program names **dapsyslog**, **dapfile**, **dapserial**, and **dapexec**, consecutively.

NOTE:: ***DAP SYSLOG is a special DAP. The entry in the configuration file is for a reference only since its implementation is hardcoded.***

Using cdata.conf to Define a Channel

The System Administrator is responsible for defining channels. The System Administrator defines the correct channels to handle the connection for the observed entities.

The **DAP** channels are defined in the **DAP** configuration file called **swa/etc/conf** sub-directory. The **cdata.conf** file contains the definition of every channel that can be referenced by **rc-ent**.

The following is an example of **cdata.conf**.

```
[com1]
chanSystem = MPRAS-Sparrow
chanDevice = /dev/term/s01
chanTerm   = vt100
chanType   = SERIAL
;----------------------
baudrate = 1200
parity   = even
charsize = 8
stopbits = 2

[com2]
chanSystem = MPRAS-Sparrow
chanDevice = /dev/cua/b
chanTerm   = vt100
chanType   = SERIAL
;----------------------
baudrate = 9600
parity   = none
charsize = 8
stopbits = 1

 [file-aaa]
chanType    = FILE
chanSystem  = plain file
chanDevice  = /tmp/swa/aaa
chanTerm    = none

 [app-date]
chanType   = APPLICATION
chanSystem = Date
chanTerm   = none
chanDevice = /bin/date

[app-ksh]
chanSystem = Korn Shell -VT100
chanTerm   = vt100
chanDevice = /bin/ksh
chanType   = APPLICATION
```

Figure 4.5 - Sample cdata.conf file.

The following table describes the keywords used in **cdata.conf**.

Keyword	Description
[LABEL]	The name of channel. The <channame> that can be referenced using **rc-ent**.
chanType	The name of DAP channel type. The name must be defined previously in **cype.conf**.
chanSystem	The short description of the system name connected through this channel.
chanDevice	The channel device name. This keyword may have different meanings, depending on the channel type. If **chanType** is **FILE**, specify a UNIX file name. If **chanType** is **APPLICATION**, specify a UNIX command/script. If **chanType** is **SERIAL**, specify a serial communication port device name.
chanTerm	The terminal emulation type. The supported terminal emulation types are listed in the **swatop/lib/term** sub-directory.

*Table 4.3 - **cdata.conf** Keywords*

The following keywords are used only in the **DAP** type **SERIAL**. This information is required to configure the serial communication port.

Keyword	Description
baudrate	The baud rate of the specified serial communication port.
parity	The parity used to check integrity of the data.
charsize	The size of character in bits.
stopbits	The number of stop bits.

Table 4.4 - DAP Type Serial Keywords

If using a **DAP** with the type **FILE**, specify a delay interval between the bursts of test data on all I/O lines ("-time <sec>" option). In a **DAP** with type **SERIAL**, a **chanEmu** option may be used to turn on terminal emulation preprocessing.

Verifying the Connectivity to the Entity

Figure 4.6 illustrates the connectivity between the *SentraWare* host, the **DAP** computer **eagle** and the monitored entity **a-sparrow**.

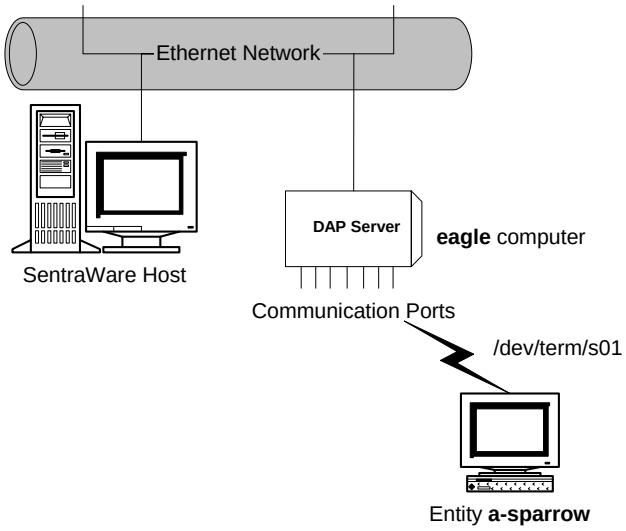


Figure 4.6 - Connectivity Between SentraWare, DAP, and a Monitored Entity.

1. At the **swash** prompt, enter `restore-log  a-sparrow`.
   This command starts the logging process for entity **a-sparrow**.

   At the **browse** session, enter a to start an **access** session.

2. Enter `b entity=a-sparrow`.

   This command starts the browser program. An entry such as "102:112 Available" is displayed, indicating that the channel has been successfully opened/restored.

   The bottom half of the screen displays the access session. The user types and gets a response back from the entity being accessed.

Controlling the *SentraWare* Processes

The **executor** module controls and monitors all *SentraWare* modules. If a *SentraWare* module crashes and its exit code is non-fatal, **executor** automatically tries to reactivate the module. Using **executor**, a System Administrator can shutdown or restart the an entire *SentraWare* module as well as control the state of individual *SentraWare* modules.

This section covers the following:

- **executor** and the related files
- **exit** codes

Description

The **executor** module is the only *SentraWare* module activated by the UNIX operating system. **executor** activates when the UNIX system goes into Multi-user state. During start up, **executor** initializes all required semaphores, queues, and shared memory segments. **executor** consolidates all system errors as the **myself** entity and applies the MA Pattern for the **myself** entity. Once started, **executor** starts other *SentraWare* modules. The startup process performs the correct sequence according the *SentraWare* module dependencies. The following table lists the software modules used by *SentraWare*.

Module file name	Module Name
executor	Executor
eventdisp	EventDispatcher
datadisp	DataDispatcher
dapmgr	DapManager
chanmgr	ChannelManager
manal	MessageAnalyzer
vtlogger	VtLogger
autoresp	AutoResponse
scheduler	Scheduler

Table 4.5 - SentraWare Software Modules

Before starting up the modules, **executor** sets up an environment specified in **/etc/sentraware**, current working directory to **$SWATOP** specified in **SWATOP** keyword in **/etc/sentraware**. It also can setup user/group IDs to run all modules (default is "**root**").

Every *SentraWare* module has its own initialization file (INI file). The *SentraWare* module uses the INI file to set up tracing and customization for the system. *SentraWare* can work without INI files using built-in defaults. The INI files are installed in **swatop/etc** sub-directory. **executor** and all other *SentraWare* modules are installed in **swatop/sbin** sub-directory.

Once started, every *SentraWare* module sends heartbeats to **executor**. If **executor** does not receive a heartbeat from a *SentraWare* module in a predetermined time, **executor** assumes that the module has stopped functioning. If an exit code is non-fatal, **executor** automatically tries to reactivate the module. The following table lists the exit codes.

	Description	Exit Code
	OK	0
	BUG	666
*	CONFIGURATION	100
	SIGNAL	101
*	WRONG_PROTOCOL	102
	RECONNECT_FAILURE	103
*	USAGE	104
*	DATABASE_ERROR	105
*	NO_PERMISSION	106
	BAD_LOGIN	107
	TIMEOUT	108
	SERVICE_REFUSED	109
	RESOURCE_UNAVAILABLE	110
*	EXEC_FAILED	111

Table 4.6 - SentraWare Exit Codes

NOTE: ***The Exit Codes marked by * are considered fatal by the executor. executor does not restart the module terminated with a fatal exit code and shuts down all modules that rely on it.***

executor Commands

output-system	Get the current status and statistic information of the specified module(s). The valid format is: `output-system [module=<modname>,...,<modname>] <t>`
inhibit-system	Terminate the specified module(s). The valid format is: `inhibit-system [module=<modname>,...,<modname>] [propagate=<y/n>]`
allow-system	Restart the specified module(s). The valid format is: `allow-system [module=<modname>,...,<modname>] [propagate=<y/n>] <t>`

Command-Line Arguments

For each command, a user can specify the following information on the command line.

module	Respond with one or more valid Module Name (<modulename>). The module name is case sensitive; only the first 3 letters are required. Refer to Table 4.2. *SentraWare* Module for the valid *SentraWare* module name. Default response: The default value is **all**.
propagate	If the value **no** (or **n**) is specified, only the specified module is restarted/terminated. If the value **yes** (or **y**) is specified, the specified module and also the dependent module(s) are restarted/terminated. Default response: The default value is **no**.

Examples

The following shows the **output-system** command used to list the current status and statistic information on all *SentraWare* modules.

```
@ output-system/
module = /  PF


                      System Status Report
                      Wed Aug 19 18:44:23 1999

      NAME       PID  STATUS MODE STARTS   LAST START    LAST TERM.    REASON
-------------- ----- ------ ---- ------ -------------- -------------- ------
EventDispatcher 3154  ALIVE  RUN       1 08-19,16:44:16 08-19,16:44:16  101e
DataDispatcher  3155  ALIVE  RUN       1 08-19,16:44:16 08-19,16:44:16  101e
DapManager      3156  ALIVE  RUN       1 08-19,16:44:16 08-19,16:44:16    9s
ChannelManager  3157  ALIVE  RUN       1 08-19,16:44:16 08-19,16:44:16  101e
MessageAnalyzer 3913  ALIVE  RUN       1 08-19,16:44:16 08-19,16:44:16   10e
VtLogger        3159  ALIVE  RUN       1 08-19,16:44:17 08-19,16:44:17  101e
AlarmProcessor  3169  ALIVE  RUN       1 08-19,16:44:17 08-19,16:44:17  101e
AutoResponse    3161  ALIVE  RUN       1 08-19,16:44:17 08-19,16:44:17  101e
Scheduler       3162  ALIVE  RUN       1 08-19,16:44:18 08-19,16:44:18  101e
PatternServer   3163  ALIVE  RUN       1 08-19,16:44:18 08-19,16:44:18  101e
UIServer        3164  ALIVE  RUN       1 08-19,16:44:19 08-19,16:44:19    0e
  OK


@
```

Figure 4.7 - Sample System Status Report

The example below shows the **allow-system** command used to restart only the Alarm Processor module.

```
@ allow-system /

module = Ala/
propagate = n/  PF


                      System Status Report
                      Fri Aug 20 11:16:07 1998

      NAME       PID  STATUS MODE STARTS   LAST START    LAST TERM.    REASON
-------------- ----- ------ ---- ------ -------------- -------------- ------
AlarmProcessor  27068 ALIVE  RUN       1 12-01,17:50:30 N/A              N/A
  OK
```

*Figure 4.8 - Sample **allow-system** Restarting the Alarm Processor Module*

Archiving

The size of the *SentraWare* log database is determined by the size of storage devices used in the system. The incoming messages cause the log database to grow. To prevent loss of log data in the log database, an automatic archiving system is included in *SentraWare*. Log data that have been successfully archived are deleted from the log database.

The messages are copied to the archive file regularly. This archive file can then be saved to tape and stored off-line. The archived file can later be restored to disk and examined, as necessary. (The archived log files may be browsed with the *SentraWare* Log Browser facility. (See the *SentraWare Reference Manual*.)

Archive Structure

A subtree in the *SentraWare* directory is maintained by the archiving system. The location of the archive subtree is defined in the **/etc/sentraware.ini** file. If not defined, the location for the archive files is in **$SWATOP/archive** sub-directory. **SWATOP** is defined in the **/etc/sentraware**.

The ARCHIVE top directory contains sub-directories with entity names taken from the sub-directories name, for example:

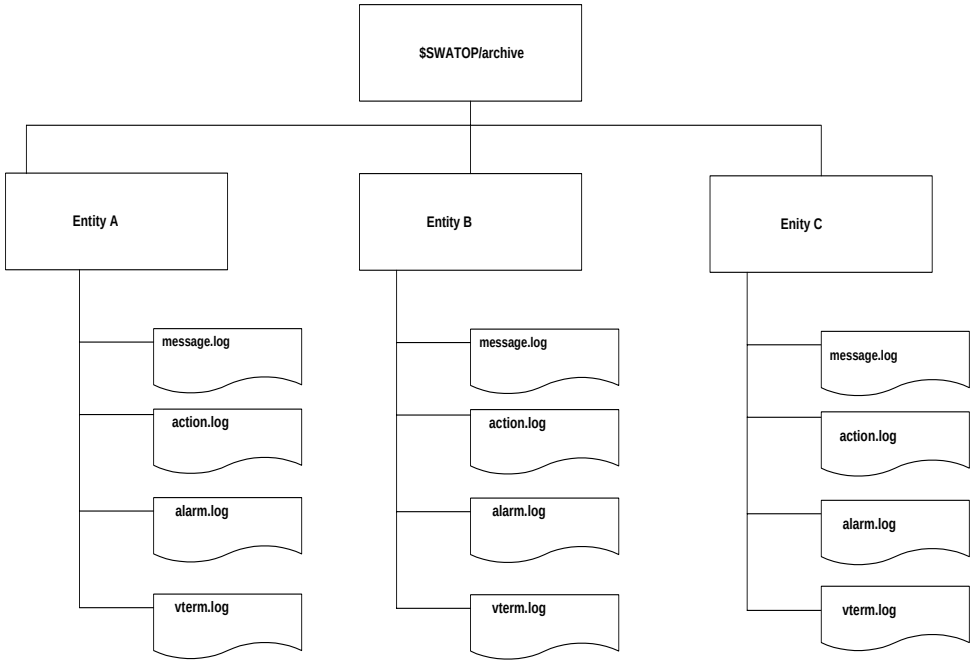


Figure 4.9 - SentraWare Directory Structure

Archiving Criteria

Archiving criteria is defined individually for each entity in the **rc-ent** command. The **logtime** option defines the number of days to store entity logs in the logging database. The default value for the **logtime** is 30 days. If the option **to** is not specified, **archive** ensures that the logging database retains records for the last 30 days only for the entity. All older records are sent to archive. To ensure all data is flushed from the database, run **archive** with the **entity** and **to** parameters, specifying the current data value.

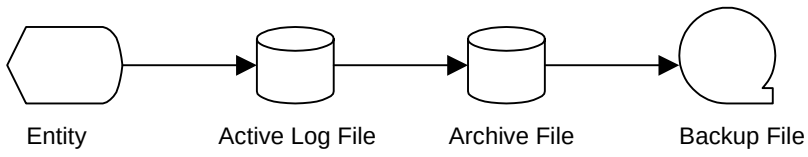
Archive Periods

When *SentraWare* is installed, a schedule called **sys-archive** is set up, scheduling **archive** to run at 4:00 a.m. every day. The archive program copies the *SentraWare* active log records from the database to the archive file. To change the start-time; see the Chapter 5, “Working with Actions and Scheduler” in this document.

Archive Backup

A successful archive overwrites the archive files. Ensure an archive backup has been made before performing a new archive operation. The archive backup program copies the *SentraWare* archive files to tape or other media, freeing up disk space.

In general, the data logged by *SentraWare* follows the path illustrated below:



Refer to your UNIX system documentation for the description of the procedures to perform the selective file backup.

Managing the *SentraWare* Host Machine

This section describes the following topics:

- Crash recovery procedures
- Configuration limits
- Monitoring internal health of the system.

Crash Recovery Procedures

A system crash, loss of power, or a complete stoppage of system operations rarely occurs. However, you must perform the following steps to recover the system when a crash occurs:

1. Review the messages on the system console to record and pursue any error messages.
2. If a power failure caused of the crash, place the switch in the "ON" position when power is available. The system boots and checks the file systems automatically.

   If the system fails to boot properly, follow the recommendations provided in the hardware documents supplied with the computer.

3. If the cause of the crash is software, a copy of the program causing the crash is usually left behind and can be examined with the program **crash**. Refer to the *System Administration Reference Manual*, for the operating system being used, for a description of how to use this program.

NOTE: ***If a system dump is desired, it must be taken before the system is booted at a normal level. The sysdump program is detailed in the UNIX System V - System Administration Reference Manual.***

   If, after rebooting, the system appears to run correctly, no further action is needed.

4. Check the **myself** entity for error messages about *SentraWare* operation.

   If error messages are displayed about certain files, correct the problem by restoring those files from incremental backup tapes. See *Recovering Corrupted Files*, in the next section. After restoring the files, if the system appears to run correctly, no further action is needed. If the system still sends error messages, perform a complete system restore.

Recovering Corrupted Files

If after a system crash, some files become corrupted, try to recover the corrupted files from the most recent incremental backup media.

In some cases, if older files are restored, rework (recompile, merge, etc.) all related files. Otherwise, the databases used by the *SentraWare* system may disagree with what users see in files they can access, and it may appear that the system is behaving erratically. If this occurs, perform a complete system restore.

Configuration Limits

The Configuration Limits shown in the following table indicate hardware and/or software factors, which limit the operation of *SentraWare*.

Peripheral/Resource	System Limit	Notes
Definable Center, Monitored Entities, Logging Channels	Unlimited	Limited by hardware and network connections.
User-Definable datasets	32	
Simultaneous Dial-Up Connections to Monitored Entities.	Up to host processor Capacity.	
Workstations	Up to host processor Capacity.	
Logging Data Storage	Limited by size of the table space defined in Oracle.	Database management.

Table 4.7 - Configuration Limitations

Load Limits

The *SentraWare* system operates in a **time-share** mode; that is, a large number of processes may be active at any time, each receiving small amounts of service from the computer, the disk, or other computer resources, as required and as available. Although the services are performed alternately; time-share mode gives users the impression of simultaneous service within the system. As higher loads are placed on the system, more processes become active and demand more system resources, contention may arise. Processes may have to wait for requested resources, and the time required to complete a task may increase. As a result, users may notice slower response to input commands and slower posting of alarms.

If the load increases greatly, data from entities can come in at a rate that exceeds the rate at which the logging and alerting processes can process the data. If this condition persists, intermediate data storage areas may fill up, causing lost data. This is called a system overload. If the entity's baud rate exceeds the specified baud rate for the line, the **DAP** closes the line down until that baud rate drops below or equal to the specified baud rate. The system automatically reopens the line.

The *SentraWare* system remains operable during overload conditions, and normal response times return when the load level is reduced. The following sections describe the factors contributing to system response time and provides guidelines for controlling the system load to prevent overloads.

Monitoring Internal Health

Check the following items regarding *SentraWare* internal health:

1. Ensure there is sufficient disk space to perform *SentraWare* functions. From a UNIX shell run the **df -k** command to determine the amount of disk space available for *SentraWare* functions..

```
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t0d0s0    1205759  460396  685076    41%    /
/proc                      0       0       0     0%    /proc
fd                         0       0       0     0%    /dev/fd
/dev/dsk/c0t0d0s3     288855   79038  180932    31%    /var
/dev/dsk/c0t1d0s7    2012390 1620689  331330    84%    /tone
swap                  828984    3496  825488     1%    /tmp
#
```

*Figure 4.10 - The **df** Command from the UNIX Shell.*

2. From the UNIX shell run the **ipcs -s** command to ensure that enough semaphores are allocated. This command also notes memory sharing. Both semaphore and memory sharing information must be present for *SentraWare* to function properly.

```
# ipcs -s
IPC status from <running system> as of Fri Jan 22 19:13:25 1999
T         ID      KEY        MODE        OWNER    GROUP
Semaphores:
s          0   0x00000001 --ra-r--r--     cams     cams
s          1   0x00000002 --ra-ra-ra-     cams     cams
s          2   0x00000003 --ra-ra-ra-     cams     cams
#
```

*Figure 4.11 - The **ipcs** Command from the UNIX Shell.*

Monitoring *SentraWare* Performance

The **$SWA/TOP/log** directory contains several log files. From a UNIX shell, examine the log files using the **vi** editor. The files contain performance information to help diagnose problems with *SentraWare*. The following example displays the content of **chanmgr-0.log**:

```
 9-18:31:56 chmApp: [ChannelManager] Created.
 9-18:31:56 chmProtoListener: [Protocol] Created.
 9-18:31:56 chmProtoEngine: [Protocol] Created.
 9-18:31:56 chmProtoManager: [Protocol] Created.
 9-18:31:56 chmEventClient: [EventDisp] Created.
 9-18:31:56 chmDataClient: [DataDisp] Created.
 9-18:31:56 chmApp: [ChannelManager] Initialize started.
 9-18:31:56 chmProtoListener: [Protocol] Initialize started.
 9-18:31:56 chmProtoListener: [Protocol] Ready to listen to port 8004.
 9-18:31:56 chmDataClient: [DataDisp] Initialize started.
 9-18:31:56 chmDataClient: [DataDisp] Initialize finished.
 9-18:31:56 chmEventClient: [EventDisp] Initialize started.
 9-18:31:56 chmEventClient: [EventDisp] Initialize finished.
 9-18:31:56 chmProtoEngine: [Protocol] Initialize started.
 9-18:31:57 chmProtoEngine: [Protocol] Trying to open entity:
          id=1001
         name="ksh"
         chan="app-ksh"
 9-18:31:57 chmEntityChan: [1001] Created.
 9-18:31:57 chmDapmgrClient: [Dapmgr] Created.
 9-18:31:57 chmDapmgrClient: [Dapmgr] Initialize started.
 9-18:31:57 chmDapmgrClient: [Dapmgr] Connecting to "localhost:8003".
 9-18:31:57 chmDapmgrClient: [Dapmgr] Initialize finished.
 9-18:31:57 chmEntityChan: [app-ksh:1001] Initialize started.
 9-18:31:57 chmEntityChan: [app-ksh:1001] Initialize finished.
 9-18:31:57 chmProtoEngine: [Protocol] Ready to log system message for entity
1001
102:101 Open started
"chanmgr-0.log" 6780 lines, 282372 characters
```

Figure 4.12 - Analyzing Performance Information.

Call Tone Technical Support at (800) 833-8663 for information about errors messages.

Archiving the Logging Data

Each entity has a log time specified individually. The log time is specified using the **rc-ent** command. The **archive** command moves all logging data (message, vterm) for the specified entities into archive files at the same time. If all arguments are omitted, **archive** places logging data for all entities in archive to satisfy **log time** criteria for every entity. For example, if an entity has **log time** specified as 30, **archive** ensures that the logging database contains records for the last 30 days for this entity, and no more. All older records are sent to archive. Messages that have been successfully archived are deleted from the logging database.

The archive file can be saved to tape and stored off-line. If necessary, the archived file can be restored later to disk and examined. (The archived log files may be browsed with the *SentraWare* Log Browser facility.)

Refer to the *SentraWare Reference Manual*, Chapter 4 for more detailed information about **archive**.

Logging Control Commands

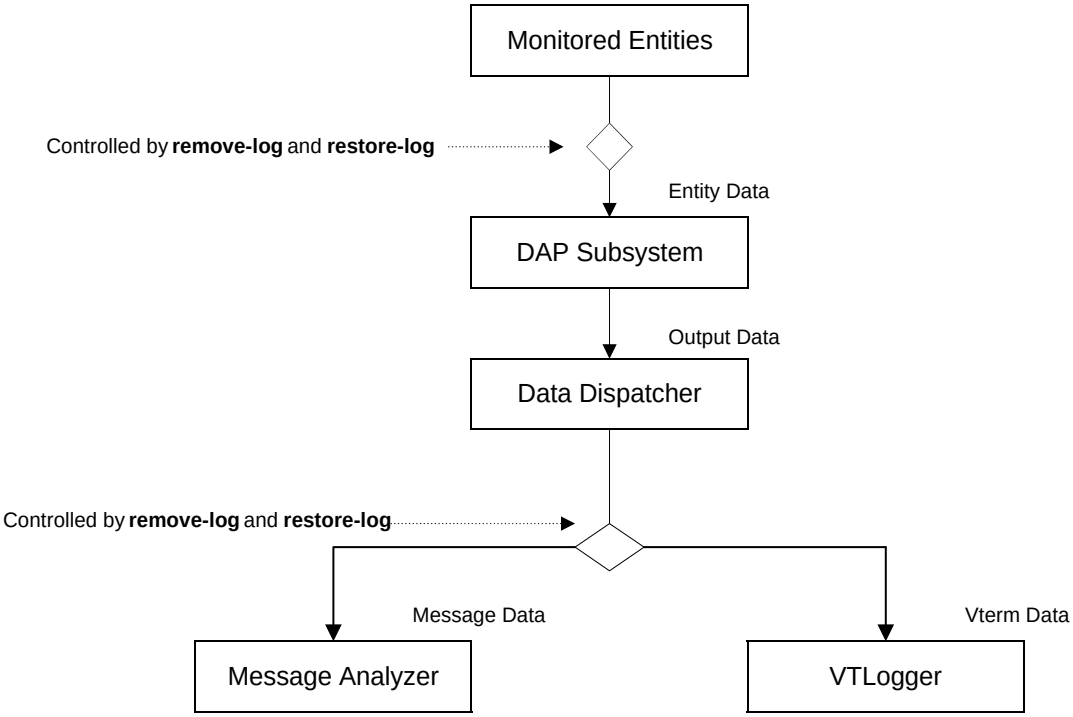


Figure 4.13 - Logging Control Commands Flow

remove-log and restore-log Commands

Entity logging is controlled by the **remove-log** and **restore-log** commands.

The **remove-log** command stops the logging process for the specified entity. The **remove-log** command places the entity logging state into "**off**" making the entity inaccessible from a *SentraWare* workstation. The **remove-log** also stops the access session(s) for the specified entity.

The **remove-log** command may be run on one or more channels for the following reasons:

1. An entity is in a state in that automatically sends messages and can become a nuisance and/or danger.

2. A software or hardware failure causes the Channel Manager to send a garbled message to the entity.

```
@ remove-log

entity = cisco-A
  PF
  OK

@
```

The **restore-log** command starts the logging process for the specified entity. The **restore-log** command places the entity logging state in "**on**", making the entity accessible from a *SentraWare* workstation.

```
@ restore-log /

entity = cisco-A /  PF
  OK

@
```

allow-log and inhibit-log Commands

The Data Dispatcher module is the *SentraWare* module that receives the output data from the entities, through the DAP system.

The **allow-log** and the **inhibit-log** commands control how entity data is processed further. The **allow-log** and **inhibit-log** prompts for the entity name and log type. The user can **allow-log** for one or more log. Refer to the *SentraWare* Reference Manual for more information.

The following allows logging for the CISCO-A entity and stops logging for vterm. The final message informs that vterm logging is already turned off for CISCO-A.

```
@ allow-log /

entity = cisco-A /
log = ?

The following log types are available:
        message
        vterm

log = message /  PF
  OK

@ inhibit-log /

entity = cisco-A /
log = vterm /  PF

VTerm Logging is already off for entity cisco-A
@
```

*Figure 4.14 - Sample **allow-log** and **inhibit-log** Execution*

output-stat-cent

Command Format

```
output-stat-cent [center=<centname>,...,<centname> <t>]
```

The output command may be abbreviated op.

Description

This command may be executed by any user to request a current status report displays for the active alarm configuration.

Outputs information for the specified centers.

The command line parameters are as follows:

center	Enter one or more (up to 20) valid center names. If **all** is specified, all centers in the system are output. Default response: The default value is **all**.

System Responses

Not applicable

Example

```
                    Center Status Report
                    Thu Nov 11 10:00:14 1999

                                                        MSG   VTERM
CENTER          LOG   ALARM INPUT MA   SCHED LOG   LOG
------          ---   ----- ----- --   ----- ----   -----
k-lucent        on    on    on    on   on    on    off
k-sknight       on    on    on    on   on    on    off
k-system75      on    on    on    on   on    on    off
pbx1            on    on    on    on   on    on    off
s-lucent        on    on    on    on   on    on    off
s-sknight       on    on    on    on   on    on    off
s-system75      on    on    on    on   on    on    off
test            on    on    on    on   on    on    off
```

output-stat-ent Command

The **output-stat-ent** command outputs the status of various system functions for the selected entities. The basic functions are logging, alarm, input message capability, message-alerting inhibition, and the scheduling status.

```
@ output-stat-ent
  PF


                        Entity Status Report
                        Fri Aug 20 14:31:53 1999

NAME         CENTER      LOG   ALARM INPUT MA    SCHED MLOG  VTLOG  AVAIL
----         ------      ---   ----- ----- --    ----- ----  -----  -----
e-siska       test        on    on    on    on    on    on    off    yes
ksh           test        on    on    on    on    on    off   on     yes
ksh1          test        on    on    on    on    on    on    on     no
myself        test        on    on    on    on    on    on    off    yes

@
```

Figure 4.15 - Entity Status Report.

Chapter 5 - Working with Actions and Scheduler

An **action** is a *SentraWare* command, or an independent program, that is executed automatically at predetermined time and date. The action must be one that can be scheduled (run in the background).

This chapter describes the following topics:

- General concept of actions and scheduler.
- Describes the creation and scheduling of *SentraWare* actions.
- Rules for creating *SentraWare* and UNIX scripts that form the basis for the actions.

Commands used to perform the above tasks are:

Command	Application
rc-act	Creates, modifies, and deletes actions used on the *SentraWare* system.
verify-act	Outputs a list of actions defined on the *SentraWare* system.
rc-sched	Creates a schedule job.
verify-sched	Displays information regarding the current schedule.
inhibit-sched	Suspends Scheduler activity.
allow-sched	Restarts Scheduler activity.

*Table 5.1 - Commands Used with **action***

NOTE: ***Consult the SentraWare Reference Manual for descriptions of the commands addressed in this section.***

Working With Actions

The following concepts are associated with automated actions:

- **action ID** - A unique character string associated with an action. **Scheduler** also references the action ID to carry out an action at predetermined date and time.
- **Auto Response module** - The *SentraWare* module that executes automation actions specified in the **Scheduler**.

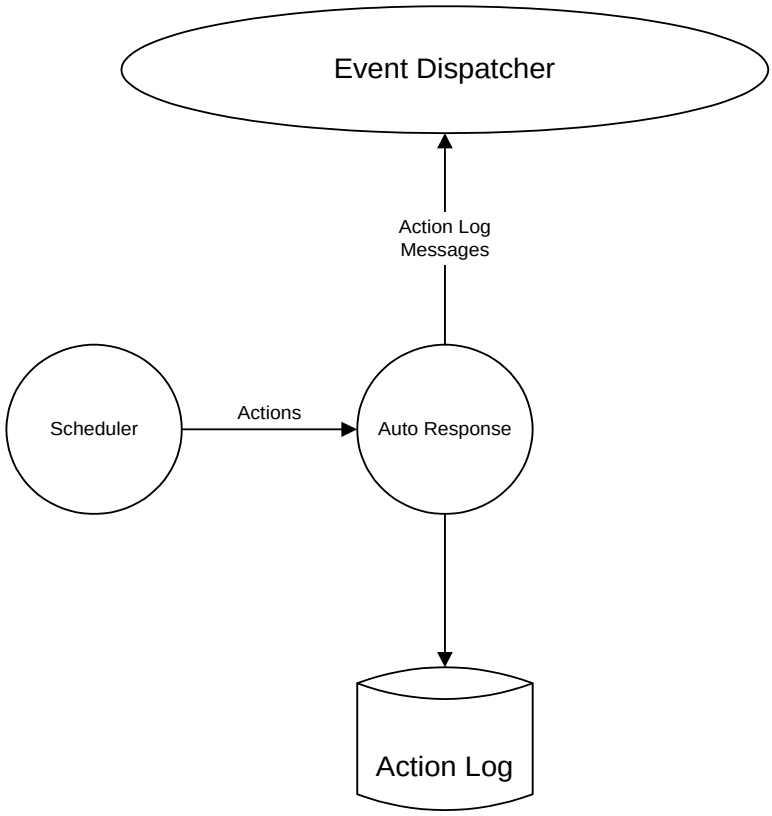


Figure 5.1 - Action Flow Diagram

When an automatic response is executed by **Auto Response**, **action logs** of the commands and execution time are maintained by *SentraWare*. The log of current actions for every entity is kept in log database named **action log**. The archived action logs are kept in the **$SWATOP/archive/<entname>** sub-directory. <**entname**> is the name of the entity.

An action is comprised of a one line command or a complex shell script consisting of a combination of several *SentraWare* and UNIX commands.

The verify-act Command

This command is used to view an automated action script defined in *SentraWare*. The command is available to all *SentraWare* system users.

The format of the command is:

```
@verify-act [action=<actname>,...,<actname>] <t>
```

The command line arguments are:

Argument	Application
action	Specifies the name of the action. When this option is selected, the data is associated with the specified action ID. The default value is **all** actions defined in the system.

*Table 5.2 - Command Line Arguments for the **verify-act** Command.*

Action Logs

The automatic action response feature maintains an action log for every entity in the log database. The action log contains current activity records for a particular entity. **archive** places logging data for all entities into archive to satisfy log time criteria for every entity. For example, if an entity has the Log Time specified as 30. **archive** ensures that the logging database contains records for the last 30 days for this entity, and no more. All older records are sent to archive.

The action log contains information for each action executed in the entity. This information includes the name of the action, the start time, completion time, the values of the action command line variables and output generated by the command, and error diagnostics generated by the various actions commands.

The action log, as well as the archived action log, may be browsed and filtered, as needed. To **browse** the current log, enter:

```
@browse entity=<entname> log=action <t>
```

Refer to the *SentraWare Reference Manual* for a description of the **browse** command.

Creating Actions

The following steps are used to create automated actions:

1. Execute the **verify-act** command to determine the actions that currently exist.

   If an action currently exists, it has been assigned by the **Scheduler**, which references the action ID.

   If an action needs to be created, the commands included in the script need to be researched to determine if they can be scheduled.

2. Use the **rc-act** command to create the actions.

3. Assign the actions created in Step 2. (Refer to “The rc-act Command.)

4. Check the **action log** and/or **archive/action.log** to determine if the action is executing properly.

The rc-act Command

The **rc-act** command is used to create actions.

When the **rc-act** command is entered, the first prompt, **rctype**, has three options: **new**, **chg**, or **out**. Enter the **rctype** selection, followed by /. *SentraWare* prompts for the following information:

action	A character string that identifies the action. The selection of the **action** requires planning. Each **action** is unique to the *SentraWare* system. The name may identify the system or the commands that are contained in the action. If the **rctype** is **out**, the action record is removed from the action database. The user is not put in the **vi** editor and no further prompts are issued.
exclusive	A flag to determine the execution mode of the action. The **yes** value makes the action run in exclusive mode, no other action execution is allowed. The **no** value makes the action run in non-exclusive mode, other actions can be executed at the same time.
priority	Priority can take any non-negative integer value. The lower the value, the higher priority. The highest priority is **0**. Default priority is **9** if the **rctype** is **new**, or the current value if the **rctype** is **chg**.
duration	Duration is the maximum time, in seconds, that the action has to finish executing. If the action has not finished execution when the duration time expires, the action is killed and an alert is generated according to the alert parameter. Default duration is **60** if the **rctype** is **new**, or the current value if the **rctype** is **chg**.
tolerance	Tolerance is the maximum time, in seconds, that the action has to start executing. If the action has not started execution when the tolerance time expires, the action is killed and an alert is generated according to the alert parameter. Default tolerance is **1800** if the **rctype** is **new**, or the current value if the **rctype** is **chg**.
alert	An alert with this name is generated if the action fails. Expiration of tolerance, duration interval, or failure of the action during execution can cause this alert. Alert name may not contain more than 20 characters. The default alert is **SWA_ACTION_ABORTED** if the **rctype** is **new**, or the current value if the **rctype** is **chg**.
dir	The directory where the action is to be executed. The default directory is **/tmp** if the **rctype** is **new**, or the current value if the **rctype** is **chg**.

After the prompts are typed and permission to execute the command has been verified, a response similar to the following appears. The user is now placed in the **vi** editing session and all system responses come from **vi** until the "Edit session completed" message appears.

```
"/tmp/eparm9343.buf" 3 lines, 17 characters
# Action body
#

~
a
restore-log entity=$ENTNAME
ESC
~
:wq
Edit session completed.
```

*Figure 5.2 - Creating a New Action Using the **vi** Editor*

The example above creates a new action, which has the **Action Body**:

restore-log entity=$ENTNAME

General Guidelines for Creating the Action Body

It is recommended that **Action Body** is used as a self-contained shell script. Since **Action Body** is stored in the database and is not stored as a UNIX file, it is best used with the fault tolerance architecture. Also, in multiple *SentraWare* host systems, another *SentraWare* system can access the **Action Body** stored in the database.

NOTE: ***The action body script is limited to 2000 characters.***

A *SentraWare* system script is a list of the UNIX or *SentraWare* commands to be executed in the order presented. These commands may be run individually. Placing them in a script in sequential order is a more efficient method of running them.

The UNIX or *SentraWare* commands used in actions must be commands that can be scheduled.

Command grouping does not allow program looping. Execution of the script proceeds from the first to last statement in the file. Command grouping offers some of the advantages of programming in general. For example, repetitive jobs may be automated and procedures may be standardized.

Parameter substitution options and comments in the scripts are provided, whereas these benefits are not available in single command mode execution outside of a script.

Parameter substitution allows use of up to 99 arguments by the script. If employed, the script file contains parameters in various positions on the command line. When the script executes, character strings replace the parameters.

Using comments in the program has no effect on the execution of the command. However, they can be an aid to the next programmer who might change the program. Properly written comments explain the original developer's intentions and describe the inner workings of sections of the program.

SentraWare provides the ability to specify variables on the command line(s) of the action. The variables include:

Variable	Application
$ENTNAME	The entity name.
$CENTNAME	The center name.
$SWATOP	The installation of *SentraWare* as defined in **/etc/sentraware**.

*Table 5.3 - Command Line Variables for the **rc-act** Command.*

Actions may be created naming the entity or center on the command line. For example, in the action shown below, the entity **comp1** is specified.

```
allow-alarms entity=comp1
```

To reduce the number of actions defined in the system, the variables listed above may be used. In the example shown below, the entity name is substituted for **$ENTNAME** before the action is executed.

```
allow-alarms entity=$ENTNAME
```

This is an action called **envtest** to describe how the automation script receives all the information from the message alerting mechanism.

Running the **verify-act action=envtest** command produces the following report:

```
                  Action Verification Report
                  Mon Jan 25 15:32:48 1999

envtest excl=no, prior=9, dur=60, tol=1800, alert=SWA_ACTION_ABORTED, dir=/tmp

#
# print out environment variables
#
echo "This is a test" 1>&2
echo "LOGNAME = $LOGNAME" 1>&2
echo "LOCATION = $LOCATION" 1>&2
echo "HOME = $HOME" 1>&2
echo "MAIL = $MAIL" 1>&2
echo "TERM = $TERM" 1>&2
echo "ENTNAME = $ENTNAME" 1>&2
echo "CENTNAME = $CENTNAME" 1>&2
echo "AUDTXT = $AUDTXT" 1>&2
echo "VISTXT = $VISTXT" 1>&2
echo "MSGDATA = $MSGDATA" 1>&2
echo "CMDLANG = $CMDLANG" 1>&2
echo "PATH = $PATH" 1>&2
```

Figure 5.3 - Sample Action Verification Report

Note that a single quote (') in the shell is equivalent to backslash and double quote ("). *SentraWare* does not process quotes, therefore, the rules are set by **sh(1)**. Also note that if you intend to use environment variables within the script, don't use arguments in the command line.

The action is invoked by the **Scheduler**. The script is run as defined in the action body, with **stdin** and **stdout** redirected to **/dev/null,** and **stderr** into a temporary file, which is later added to the action log.

The following report is produced (use "browse entity=<entname> log=action " command):

```
File  View  Jump  Search  Options                                        Help
Entity: cisco-A (action)                    Mon Jan 25 16:15:00 PST 1999
>************** Action envtest started **************
>This is a test
 LOGNAME = dmt
 LOCATION =
 HOME = /tone/personal/dmt
 MAIL = /var/mail/dmt
 TERM = xterm
 ENTNAME = cisco-A
 CENTNAME = test
 AUDTXT =
 VISTXT =
 ALMLVL =
 MSGDATA =
 CMDLANG =
 PATH = /usr/sbin:/usr/bin:/project7/scripts
>****************** Action finished *****************
```

Figure 5.4 - Sample Test Action Verification Report

Using the UNIX Shell to Run *SentraWare* Commands

If the user's UNIX log-in name is listed as an equivalent user, *SentraWare* commands can be executed directly from the UNIX prompt. See Chapter 3 - "User Administration" for information about creating an equivalent user.

SentraWare commands can be executed directly from the UNIX prompt if the environment variables "**SWAUSER**" and "**SWAPASSWD**" are set to contain the *SentraWare* user name and password respectively. The System Administrator must set other environment variables such as "**PATH**" , "**LD_LIBRARY_PATH**" and **"ORACLE_HOME"** to include the directory where *SentraWare* system commands reside. This script may be inserted in the **.profile** file.

UNIX System Pathnames to *SentraWare* System Commands

SentraWare commands can be executed from the UNIX shell. The *SentraWare* commands are called from the files where they reside in the system. The commands are organized in one directory, **$SWATOP/usr/bin**.

The following is a sample list of commands with complete paths:

```
/tone/swa/usr/bin/access
/tone/swa/usr/bin/allow-log
/tone/swa/usr/bin/allow-system /tone/swa/usr/bin/archive
/tone/swa/usr/bin/bin
/tone/swa/usr/bin/browse
/tone/swa/usr/bin/swash
/tone/swa/usr/bin/filter
/tone/swa/usr/bin/inhibit-log
/tone/swa/usr/bin/inhibit-system
/tone/swa/usr/bin/output-stat-ent
/tone/swa/usr/bin/output-system
/tone/swa/usr/bin/playback
/tone/swa/usr/bin/rc-act
/tone/swa/usr/bin/rc-cent
/tone/swa/usr/bin/rc-ent
/tone/swa/usr/bin/rc-sched
/tone/swa/usr/bin/rc-usr
/tone/swa/usr/bin/restore-log
/tone/swa/usr/bin/remove-log
/tone/swa/usr/bin/verify-access
/tone/swa/usr/bin/verify-act
/tone/swa/usr/bin/verify-cent
/tone/swa/usr/bin/verify-ent
/tone/swa/usr/bin/verify-perm
/tone/swa/usr/bin/verify-sched
/tone/swa/usr/bin/verify-usr
```

Figure 5.5 - SentraWare Commands List in UNIX

SentraWare commands executed from a UNIX system environment are called directly using the full UNIX system path name, any required arguments, and the proper terminator.

UNIX System Mode Command Syntax

When *SentraWare* commands are used from a UNIX system user interface, normal syntax rules are changed. The environment becomes a UNIX environment, therefore, syntax similar to UNIX system syntax is applied on top of the normal *SentraWare* command syntax.

Syntax rules differ, depending on whether:

- The *SentraWare* command is called with all arguments specified on the command line with no input from the user.
- The *SentraWare* command is called so additional input from the user is required.

Syntax Rules with All Arguments Specified on the Command Line

The following rules apply when the *SentraWare* command is called with all desired options specified on the command line with no input by the user.

- The command line terminator must be **ENTER** (return). If the terminator is applied directly after / or **#,** prompting or help is displayed. For example, if the **verify-usr** command is called from a UNIX system program, the call looks similar to the following:

```
/tone/swa/usr/bin/verify-usr <terminator> <enter>
```

  If the forward slash (/) terminator is used, the command prompts for additional information. If the question mark (**?**) is used, the required syntax from a UNIX system perspective is displayed.

- When commands and arguments are supplied on the command line, the command word and first argument must be separated by a blank space, an equal sign (=) character, and another blank space. If more than one argument is supplied, the remaining arguments are separated by a blank space. In the following example, the call to **verify-usr** requests information about users **dan**, **mary**, and **east**.

```
/tone/swa/usr/bin/verify-usr user = dan mary east
```

If the **op-stat-ent** command is used, the line in the program is:

```
/tone/swa/usr/bin/output-stat-ent
```

If the command output is redirected to file, it resembles the following:

```
/tone/swa/usr/bin/output-stat-ent > statent
```

If arguments are supplied for the keywords, the command line in the program resembles the following:

```
/tone/swa/usr/bin/verify-usr user = paw ajw east
```

Notice that the keyword and the first argument are separated by a blank space followed by an equal sign (=), then another blank. When multiple arguments are provided, they are separated by a single space.

As another example, a file called **sysusr.sh** is developed to take user names as arguments, and display the *SentraWare* system information about each user. The file may contain the following line:

```
/tone/swa/usr/bin/verify-usr user = $1
```

The **$1** is the UNIX system parameter substitution variable. The first argument supplied on the UNIX system command line replaces the **$1** before the command is executed. (For further details, consult your UNIX shell programming documentation.)

Input and Output Redirection

UNIX system input and output redirection rules apply, although some *SentraWare* commands may not permit one or the other. Consult the description of the command in the *SentraWare Reference Manual* to determine what form of redirection is allowed. In the following example, output from the *SentraWare* command **verify-usr** is placed in a file called **list**.

```
/tone/swa/usr/bin/verify-usr > list
```

UNIX pipe mechanisms may also be used for the redirection. In the following example, information is printed only about users who have ACCESS permission.

```
/tone/swa/usr/bin/verify-perm | grep ACCESS
```

Input Redirection to the swash Shell

Input redirection can be applied to the **swash**. The *SentraWare* commands to be executed from a UNIX system shell program have the same syntax as if typed in the **swash** shell. A file called **temp.sh** may resemble the following:

```
swash << end
op-stat-ent
bin date

end
```

If input is redirected to the **swash** shell from a file, input is for the **swash** shell use only. The file's contents cannot be used as parameter values for any *SentraWare* commands executed by the **swash** shell. All *SentraWare* commands are executed on the command line without interactive prompting.

SentraWare Scheduler

The **Scheduler** is a continuously running background program that instructs the Auto Response module to execute the specified actions on the specified entity at the specified time and in the specified directories. It provides the following advantages over entering commands manually:

- It does not "forget" to execute the commands.
- Time to manually execute the programs at the specified times is saved.
- No workstation is required to manually execute the commands.
- A record is kept of the commands and the execution times.

Jobs may execute hourly, daily, monthly, or once a year.

Only the System Administrator can create a **Scheduler** job using **rc-sched** command. As a System Administrator planning a command list for the **Scheduler**, consider the following items:

- Is the time for the **Scheduler** job during a period of low system activity?
- What is the load on the *SentraWare* system at the scheduled time?
- Can the selected commands be placed on the **Scheduler**?
- Are any other jobs on the **Scheduler** running at the selected time?
- Are the selected commands important to the efficient operation of the *SentraWare* system?
- Are the selected commands and their output relevant to your job?
- Would the job, if run from a terminal, "tie up" a terminal for an extended amount of time?

When created using **rc-sched**, a **Scheduler** job is not associated to any entities. Later using **rc-ent**, the **Scheduler** jobs can be associated with an entity. A **Scheduler** job can be associated to more than one entity. An entity can be associated to more than one **Scheduler** job.

Schedule Administration

Only the System Administrator with ADMIN permission may create a schedule file. Jobs are prioritized on the *SentraWare* system using the numbers zero (0) through nine (9), with zero having the highest priority.

Exercise care when making adjustments to the *SentraWare* system clock. Changes in the system time may create problems, depending on the severity of the change. The following are some of the reactions of the **Scheduler** to time changes:

- If the system time is moved **back more than 60 minutes**, the scheduled job is re-executed, since the **Scheduler** adjusts its timing mechanism to match the system time.
- If the system time is **moved back 60 minutes or less**, the Scheduler waits until the system time is equal to the **Scheduler** time to execute new jobs.
- If the system time is **moved forward 60 minutes or less**, the Scheduler executes all jobs scheduled between the **Scheduler** time and the system time, until the **Scheduler** time catches up to the system time.
- If the system time is **moved forward more than 60 minutes**, the **Scheduler** time is set to equal the new system time. Jobs scheduled to run between the old system time and the new system time are skipped.

Schedule jobs can be run in a specified directory. The Auto Response makes the named directory the working directory for the scheduled job.

To determine the status of the **Scheduler**, browse the Auto Response logging file **action log**. Refer to "Action Logs" in this chapter for more descriptions of an Action Log.

Examine the action log regularly to determine if scheduled jobs are executing properly, if the execution time is reasonable, or if job execution is being consistently delayed. If there are problems, take corrective steps; for example, change the scheduled time.

Scheduler Commands

The following commands create a **Scheduler** job and monitor the **Scheduler**:

Command	Application
rc-sched	Creates a schedule job.
verify-sched	Displays information regarding the current schedule

Table 5.4 - Commands to Create Scheduler Jobs and Monitor the Scheduler.

Creating a Scheduler Job

The **rc-sched** command is used to create, change or delete a **scheduler job**. A **scheduler job** defines an action to be executed in a determined time specification.

The **rc-sched** command prompts for all information not entered in the command line arguments. The prompting scheme is as follows:

rctype	Respond with one of the following keywords to specify the type of reference change: **new** - new schedule being created. **chg -** change an existing schedule **out** - existing schedule being deleted. (These names may be abbreviated **n** and **o**.) Default response: There is no default value.
schedule	A valid schedule name must be entered using a maximum of 15 characters. The name must be alphanumeric (including a hyphen (-) and must begin with an alphabetic character. If **new** is entered as the **rctype**, the name given must not correspond to a schedule that already exists in the database. If **out** is entered as the **rctype**, the schedule record is removed from the schedule database. There are no further prompts issued. There is no default value for schedule name.
action	Specify the name of the action to be scheduled. The action name entered must be an existing name created by using **rc-act**. There is no default value for a action name.
month	Respond with the schedule specification, which may include: +<n> - repeat every <n> of months. <n>,...,<n> - specific month. * - repeat every month. where <n> is any positive number. There is no default value.

*Table 5.5 - **rc-sched** Prompts (Part 1 of 2)*

day	Respond with the schedule specification, which may include: +<n> - repeat every <n> of days. <n>,...,<n> - specific day. * - repeat every day. where <n> is any positive number. There is no default value.
weekday	Respond with the schedule specification, which may include: +<n> - repeat every <n> days during the week. <n>,...,<n> - specific weekdays. * - repeat any weekday. where <n> is any positive number. There is no default value.
hour	Respond with the schedule specification, which may include: +<n> - repeat every <n> hours. <n>,...,<n> - specific hours. * - repeat every hour. where <n> is any positive number. There is no default value.
minute	Respond with the schedule specification, which may include: +<n> - repeat every <n> minutes. <n>,...,<n> - specific minutes within an hour. * - repeat every minute. where <n> is any positive number. There is no default value.

*Table 5.5 - **rc-sched** Prompts (Part 2 of 2)*

The following is an example **Scheduler** session:

```
@ rc-sched

rctype = n/
schedule = verAccess/
action = verAccess/
month = ANY/
day = ANY/
weekday = ANY/
hour = ANY /
minute = +15 /  PF
  OK

@
```

Figure 5.6 - Sample Scheduler Session

The example above shows the **rc-sched** command to create a new schedule called **verAccess**. This schedule calls for an action **verAccess** to be executed every 15 minute.

The verify-sched Command

The **verify-sched** command permits viewing of the **Scheduler** using a variety of options. The following arguments can be used to view the **Scheduler**:

Argument	Application
schedule	Displays the **Scheduler Jobs** of the specified **Schedule** name.
action	Displays the **Scheduler Jobs** of the specified **Action** name.

*Table 5.6 - Arguments for the **verify-sched** Command.*

Index

E

F

H

I

L

M

N

O

P

R

S

T

U

V

W